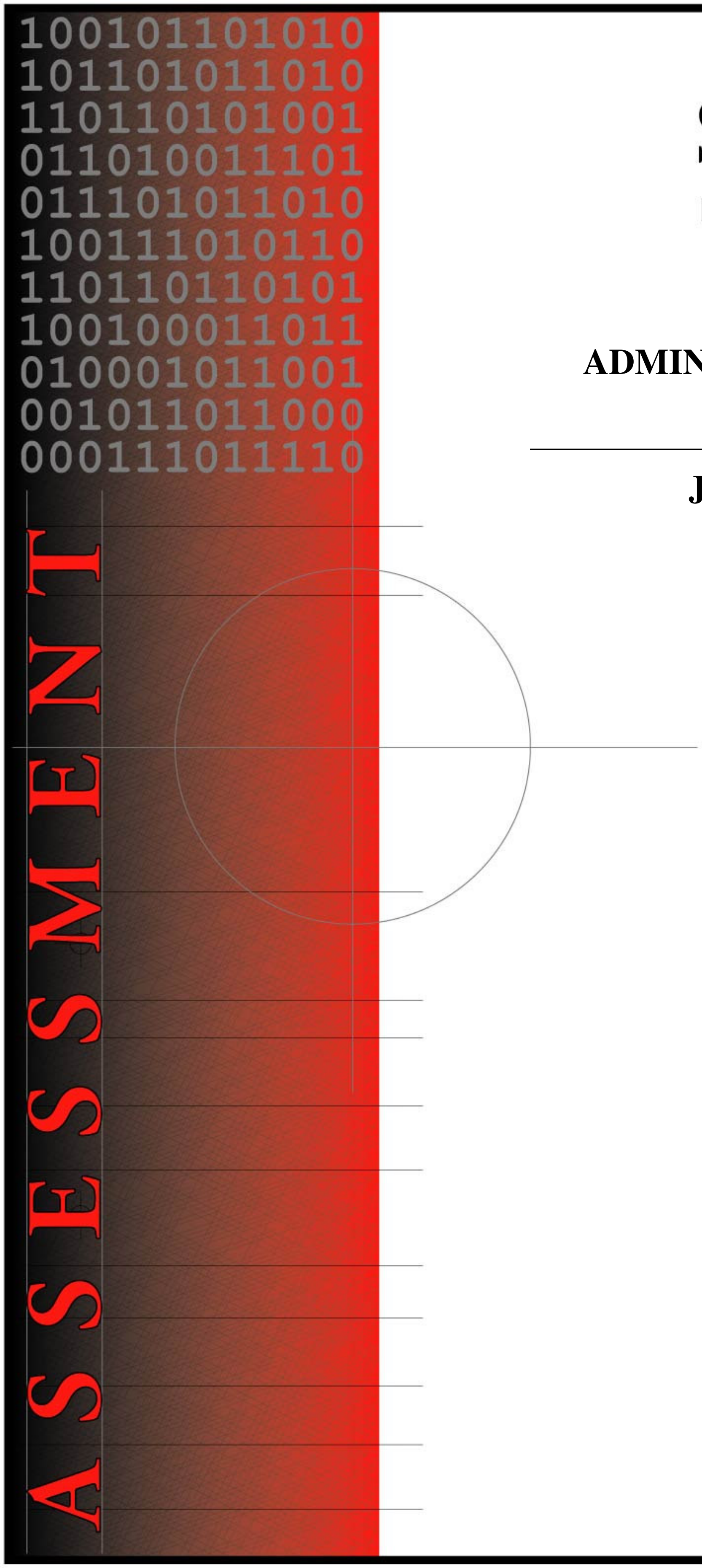


SolutionsIQ®
PROFESSIONAL SERVICES
Where ideas become action®

ADMINISTRATIVE OFFICE OF THE COURTS

JIS Architecture Assessment

Submittal Date
September 1, 2004

Prepared For
Dan Sawka
AOC JIS Development Manager

Prepared By
Will Iverson
Practice Manager
425.451.2727 x2562
425.558.7828 - fax
WIverson@SolutionsIQ.com

Julia Francis
Enterprise Solutions Manager
State & Local Government
425.519.6718 - direct
360.539.1772 - Olympia
JFrancis@SolutionsIQ.com

www.SolutionsIQ.com

TABLE OF CONTENTS

ENGAGEMENT OVERVIEW

The Administrative Office of the Courts (AOC) has embarked on an effort to integrate and replace several disparate judicial applications. The unified software application resulting from the current migration project will be the new Judicial Information System (JIS) for Washington Courts. The process envisioned is to consolidate most recently built JAVA applications (ACORDS-Appellate Court Record & Data System and CAPS-Court Automated Proceeding and Scheduling) as the basis for the new JIS. In the original migration plan, AOC planned to have this phase completed by June 30, 2005 before incorporating the requirements from other legacy applications into this new JIS application.

Over the past three years the approach to implementing the new JIS has been impacted by significant change. As a result of this impact, AOC is now taking into consideration revisiting the original migration plan, assessing the current JIS architecture, or potentially evaluating other options of approach to meet the targeted delivery dates. SolutionsIQ has been involved on a consulting basis on the AOC JIS Development Project since March, 2003. Our organization has provided services that range from architecture and design to refactoring and development and is highly knowledgeable of the core applications and technology involved in this effort.

For the purpose of this assessment, SolutionsIQ was engaged to:

- Review the current architectural design to determine the long term (20 year) viability and supportability and to make recommendations for changes or additions
- Determine the feasibility of using the ACORDS and CAPS applications in the described architecture
- Estimate the effort to migrate the two applications into the new architecture as a single application.

The desired outcome of our assessment was to identify and:

- Recommend changes or additions to the current AOC architectural design.
- List changes necessary to bring the ACORDS and CAPS applications into a proposed architecture.
- Estimate the effort to refactor ACORDS and CAPS into a proposed architecture.
- Estimate the effort to consolidate ACORDS and CAPS into a singular application.

At a high-level, our consultant team completed an in-depth analysis of the ACORDS, CAPS, and JIS Architecture and Applications. The analysis took into account both the client and server side architecture, source lines of code, complexity of features and functionality, the ability to add features, and the maintainability and long-term viability of the applications and architecture.

In the following sections we detail our analysis of each core application and its impact to the JIS Migration Plan. Following our analysis are our recommendations of architecture and project approaches to complete the JIS Migration Plan. We have included in our assessment supporting documents and diagrams outlining our processes, analyses, and findings. We believe there are options and approaches for this project that will support the AOC's vision of producing and maintaining an efficient and effective operation for the Washington State judicial system.

ANALYSIS OVERVIEW

Each of the three existing systems (ACORDS, CAPS, and JIS) were analysed for code size (both lines of code and number of files) and complexity. A review of the existing development process and developer-identified problem areas was performed. The full results of this review, includes in-depth technical information, can be found in appendices A, B and C. Appendix D compares the various tools and systems, and also compares certain technical components.

Next, the assessment team modeled the effort required to add a single create, update, retrieve, and delete of a single record for each system from scratch. This includes modeling the creation of the Swing user interface, a web-based interface, and a web service for each system. In addition, the team modeled the effort required to add a single field to a record and propogate this change throughout the system. The results of this analysis are contained in Appendix E.

Based on the analysis as recorded in Appendices A-E, the team developed a streamlined architecture, based on a view of the JIS development model but geared towards dramatically increasing developer productivity by removing unneeded architectural components. This next generation JIS architecture (dubbed JIS NG) was also modeled using the same tasks as the existing systems, with the results also included in Appendix E. The technical details of the JIS NG system can be found in Appendix F.

A desire was expressed to identify certain areas which could benefit from a focused effort in the near-term (nine months or less). Some of the potential areas identified are listed in Appendix G.

Appendix H details some of the skill sets and team roles needed based on the assessments and recommendations.

The remainder of this summary links the findings to an effort driven model for the conversion of the applications to the JIS NG architecture.

It is important to note that this analysis was conducted from a strict technical architecture perspective. Potential savings (or additional costs) which may arise from functional changes (including merging of functionalty) were not considered, nor the costs (or savings) associated with release cycles demands or other project planning related components. Finally, no effort associated with decision making at an executive level was considered.

ACORDS ANALYSIS

A detailed technical view of the ACORDS system is contained in an attached appendix. The following key points are of interest to business owners:

- The current ACORDS system is regarded as quite unmaintainable. As shown in Figure 1, despite the fact that the application contains over 1,000 source files, only 39 files account for over 50% of the application code.
- This complexity, arising from both architectural problems as well as inefficient programming, makes it extremely difficult to add new functionality or perform maintenance.
- In addition, this complexity makes it very difficult to perform testing of the application.

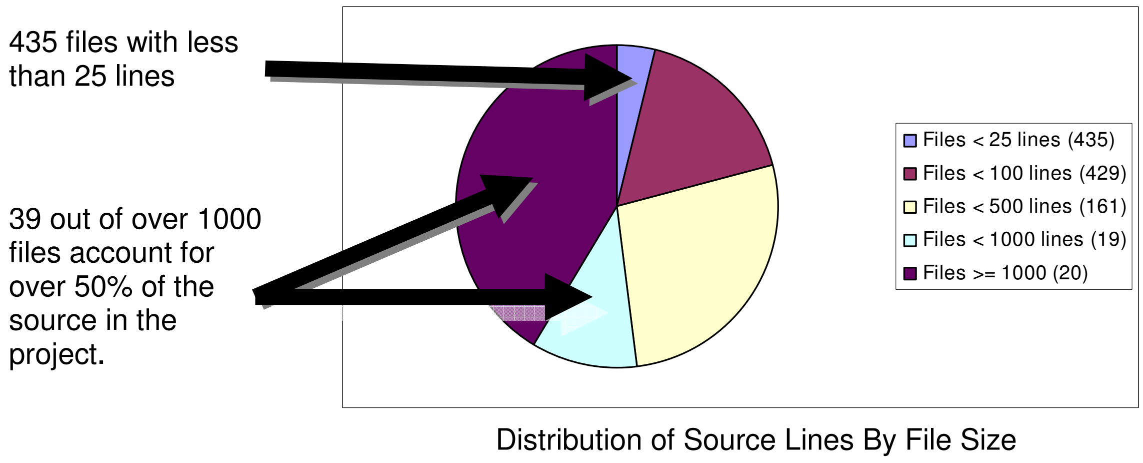


Figure 1: ACORDS Source Breakdown

Effort required to implement a new create, update, retrieve, and delete operation (as described by Appendix E) using:

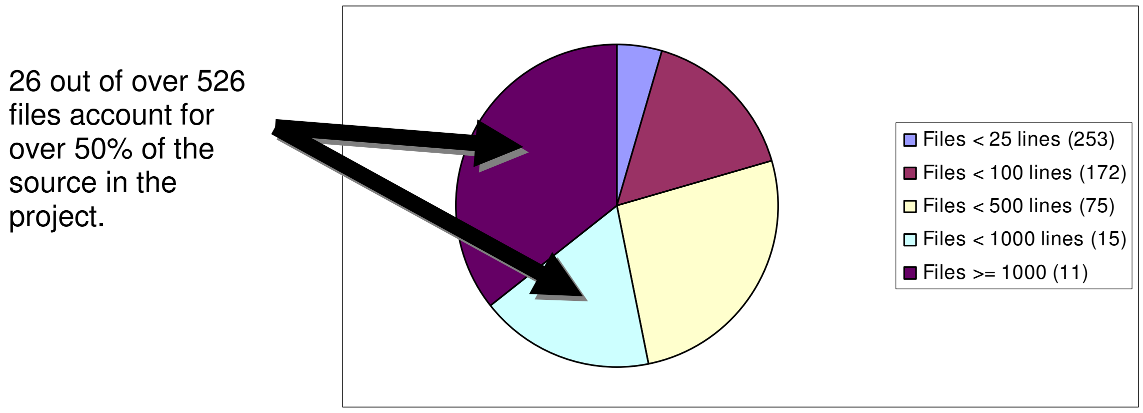
Swing	23 days
Web	25 days (estimated)
Web service	26 days (estimated)

Effort required to add a new field to a table, propogated through the system to user (as described by Appendix E): 9 days.

CAPS ANALYSIS

A detailed technical view of the CAPS system is contained in an attached appendix. The following key points are of interest to business owners:

- The current CAPS system is regarded as unmaintainable. As shown in Figure 2, despite the fact that the application contains over 500 source files, only 26 files account for over 50% of the application code.
- CAPS is in a better situation than ACORDS largely due to a smaller overall application – the profile, in terms of code and time required to implement new functionality, is otherwise largely similar to ACORDS.
- The complexity of CAPS, arising from both architectural problems as well as inefficient programming, makes it extremely difficult to add new functionality or perform maintenance.
- In addition, this complexity makes it very difficult to perform testing of the application.



Distribution of Source Lines By File Size

Figure 2: CAPS Source Breakdown

Effort required to implement a new create, update, retrieve, and delete operation (as described by Appendix E) using:

Swing	20 days (estimated)
Web	18 days
Web service	24 days (estimated)

Effort required to add a new field to a table, propogated through the system to user (as described by Appendix E): 3.75 days

JIS ARCHITECTURE ANALYSIS

A detailed technical view of the JIS system is contained in an attached appendix. The following key points are of interest to business owners:

- The current JIS system is much more maintainable than CAPS and ACORDS. As shown in Figure3, while the total application in terms of files is on par with ACORDS, the files are much smaller and more maintainable.
- While JIS is in a much better state than CAPS and ACORDS, it was determined that the application still included additional architectural components and layers unneeded for (and interfering with) the delivery of required business functionality.
- These unneeded architectural components, in addition to increasing the time needed to implement functionality, also increase the risk of failure, the time needed to train staff, testing costs, and more.

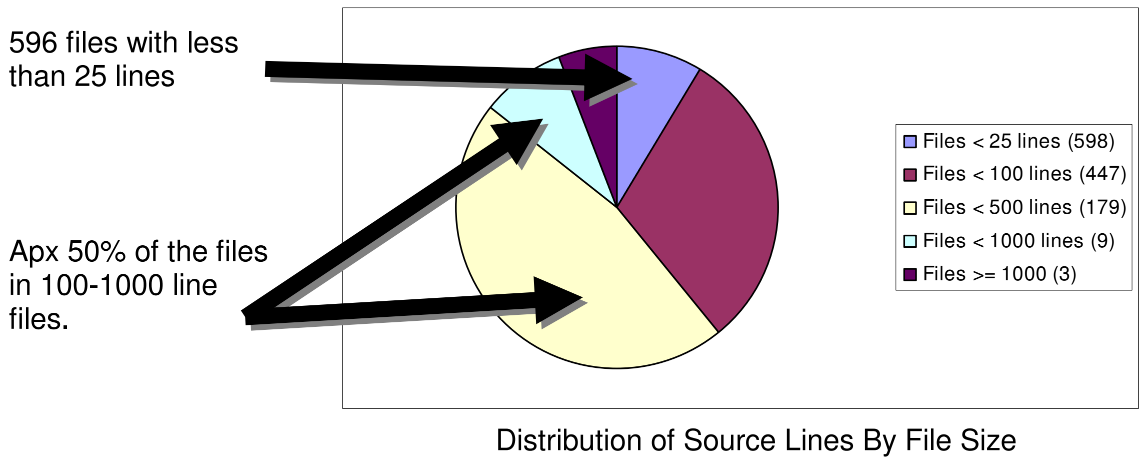


Figure 3: JIS Source Breakdown

Effort required to implement a new create, update, retrieve, and delete operation (as described by Appendix E) using:

Swing	21.5 days
Web	19 days (estimated)
Web service	24.5 days

Effort required to add a new field to a table, propogated through the system to user (as described by Appendix E): 6 days

ARCHITECTURE ASSESSMENT OVERVIEW

Given the limitations of the CAPS, ACORDS, and JIS systems, the recommendation is to move to a streamlined "next generation" version of the JIS system, hereafter referred to as JIS NG.

The JIS NG development model is intended to be highly productive and familiar to a typical Java developer. After an initial (est. 3 calendar months) development phase, all new functionality will be implemented on the new architecture. At this time other migration efforts will be started to move the business functionality of ACORDS, CAPS, and JIS onto the JIS NG architecture.

Services such as Calendaring and Case Management would be implemented on JIS NG in a way that could be used by all court levels. Depending on the functionality, application development will merge and share the various business functionality components and/or allow each court level to access their own JIS NG system through a common interface. Over time these separate backends would be merged.

Using the same model for calculating effort required to implement a new create, update, retrieve, and delete operation as described above and in Appendix E) using the appropriate user interface, the effort to implement new features on JIS NG is estimated at:

Swing	11.5 days
Web	11 days
Web service	19.5 days

Similarly, the effort required to add a new field to a table, propogated through the system to user (as described by Appendix E) is estimated at: 2.5 days.

This represents a significant savings, allowing for more efficient delivery of new functionality and reduction in maintenance.

To provide for an incremental migration, reusable components from existing legacy systems will be removed and (where appropriate) generalized for multiple court levels. It would reuse as much of the existing code and functionality of CAPS and ACORDS as possible. As new components such as Case Management become available, legacy users would be shifted to the new system as seamlessly as possible.

Migration Effort Overview

The migration would consist of two phases: a planning and upfront design phase of approximately three calendar months, and a second phased migration phase.

The planning phase affords three teams, in parallel, for three months to implement the necessary architectural design and improvement work to address needs as described in the appendices.

Phase 1

Phase 1 consists of three teams, one to address a unified persistence model (i.e. Java application integration with the database), a team to attempt to refactor (from a design perspective) the unmaintainable files in ACORDS, CAPS and JIS as identified above, and finally a third team to fully document and build the JIS NG development platform as well as demonstrate the JIS NG platform effectiveness by delivering a component of business value.

Phase 1A: Unified Persistence

This team will develop a unified persistence model for JIS NG. This single coherent persistence view will include O/R and database view recommendations, including the generation of an O/R view of a significant portion of the database. A senior DBA will work with this team to identify potential areas of concern, assist with performance analysis, and profile the existing applications for poorly performing components. The expectation is not to make changes which will significantly perturb the existing production database, but to ensure that the application development and database management designs are performed in conjunction.

Phase 1A Effort: six person team for 3 months. 2 Architects, 1 Senior DBA, 1 Developer, 1 QA, 1 PM.

Phase 1B: Design Refactor

This team will perform a design refactor of the existing application source files of 1,000 source lines of code and greater. This includes 39 files in ACORDS, 26 files in CAPS, and 3 files in JIS. This is necessary for future maintainability, testing, and any attempt to generalize or otherwise reuse components of these applications. Regardless of any new target architecture, this effort is required to effectively move these systems forward.

- This team may contribute the results of this refactoring back to the maintenance teams assigned to ACORDS, CAPS, and JIS, but is not expected to build production-ready code. It is important to keep in mind that these few files represent over 50% of the source for ACORDS and CAPS.

Phase 1B Effort: Eleven person team for 3 months. 1 Architect, 4.5 Developer, 4.5 QA, 1 PM, broken out as 2 Dev/1.5 QA for ACORDS, 1.5 Dev/1.5 QA for CAPS, and 1 Dev/1 QA for JIS.

Phase 1C: JIS NG

This phase is intended to build and document JIS NG development environment, including pulling together tools, setting up the repository, identifying functional silos, and establishing viability by implementing an initial business goal.

Phase 1C Effort: Fifteen person team for 3 months. 2 Architects, 1 Senior DBA, 10 Developers, 1 QA, 1 PM

Phase 2: Migration

The full architecture for the JIS NG project is described in an appendix. From a business user perspective, in addition to improvements intended to help drive rapid application development, the JIS NG project moves from a model which describes monolithic applications to a service centric model, as shown in Figure 4. Each service can be updated and maintained by an independent team, publishing interfaces usable by other teams.

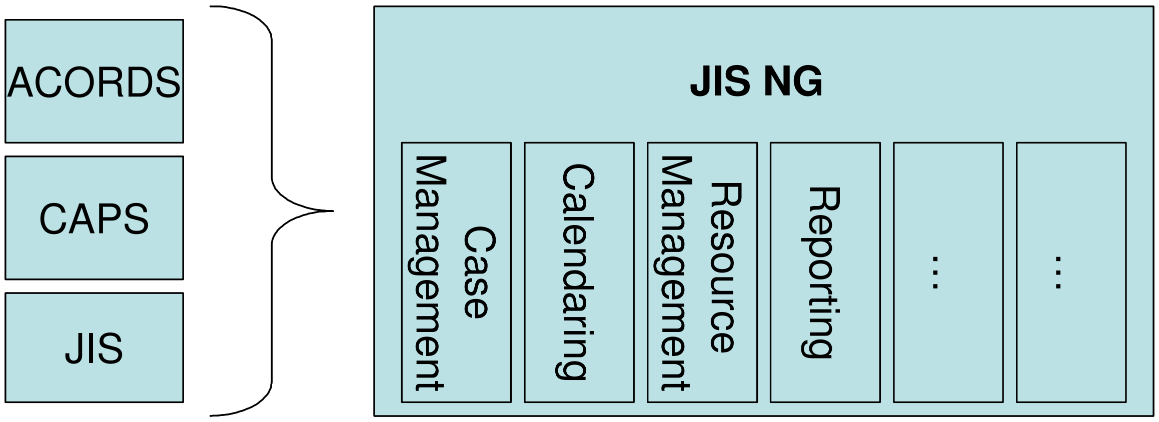


Figure 4: From Monolithic To Services

As components are migrated to JIS NG, web services and JINI services can be used to access JIS NG functionality removed from ACORD/CAPS/JIS and placed into JIS NG.

Phase 2 Effort Assessment

The following effort assessment is based strictly on an architectural/developer-centric view of the application – the potential for significant savings by merging existing functionality is not specified, as no end user analysis of benefit has been performed. Exposing merged functionality (including incremental additions to the user interface, backend functionality merging, etc.) are NOT included in the scope of this effort analysis.

Therefore, this analysis is performed from the perspective of a strict "port" – moving the systems from the respective current architectures to the new systems.

First, an attempt was made to convert the various applications to a single consistent measure, a JIS source file, as shown in Table 1.

Application	**Files**	**Relative Complexity***	**Weighted Files**
ACORDS	1141	2.05	2333.9
CAPS	646	1.86	1204.8
JIS	1237	1.00	1237.0

***based on lines of code/file as compared to JIS**

Table 1: Consistent Source File Measure

As noted above, the effort required to implement a new feature for each architecture is provided. After weighting the file count to the JIS environment, an assumption was made (based on the JIS development profile) that seven files account for a single "feature." Based on this model, the following number of "features" can be found across the following applications:

Application	Weighted File Size	Est. JIS -> JIS NG Features
ACORDS	2333.9	333.4
CAPS	1204.8	172.1
JIS	1237.0	176.7

Table 2: Estimated Features/Application

Given 11.5 days to implement a JIS NG Swing feature and 11 days to implement a Web feature, this works out to 373.1 developer months to convert all three systems to JIS NG.

Application	JIS NG Features	JIS NG Effort (dev months)
ACORDS	333.4	184.3
CAPS	172.1	91.0
JIS	176.7	97.7

Table 3: Estimated Migration Development Effort

Obviously, project management and quality assurance estimates must be included. Given a standard ratio of one (1) developer to three-quarter (0.75) test and one-quarter (0.25) project management resources, this results in a total effort of 746.1 months, excluding ramp time.

To fully model this, three base scenarios have been developed, the first based on a (purely theoretical) nine (9) month delivery, a twenty-one (9+12=21) month delivery, and a thirty-three (9+24) month delivery.

Given a 9 month delivery (high risk, theoretical only), the head count required would be 129.3. Considering that the first three months would be consumed by training, staffing, etc, this would result in a total effort of 1164.0 months. This is not viewed as a realistic scenario, given the risk and cost associated with this effort.

Given a 9+12 month delivery, a total of 43.1 head count would be required. Again, considering the training & staffing for these resources, this evaluates to a total effort of 905.3 months.

Finally, given a 9+24 month delivery, a total of head count of 25.9 would be required, for a total effort of 853.6 months.

Note that these effort assessments for phase 2 are in addition to the effort assessments for phase 1.

Maintenance Comparison

As a final analysis, a look at the costs associated with incrementally adding to the existing functionality of ACORDS, CAPS, and JIS was performed. As described above, a new feature and incremental feature have been modeled for each platform.

First, assume 150 new features (such as the ability to insert, retrieve, update and delete a single record) and 300 incremental features (such as adding a field to an existing table) for each of the three systems (450 new features and 900 incremental features total). Note that a significant portion of these new and incremental features may actually represent bug fixes to the existing application.

Given these feature counts, on the existing systems this represents 6,150 days of effort for ACORD, 3,825 for CAPS, and 3,675 for JIS. Including test and project management resources (per the formula above), this represents an effort of 27,300 days.

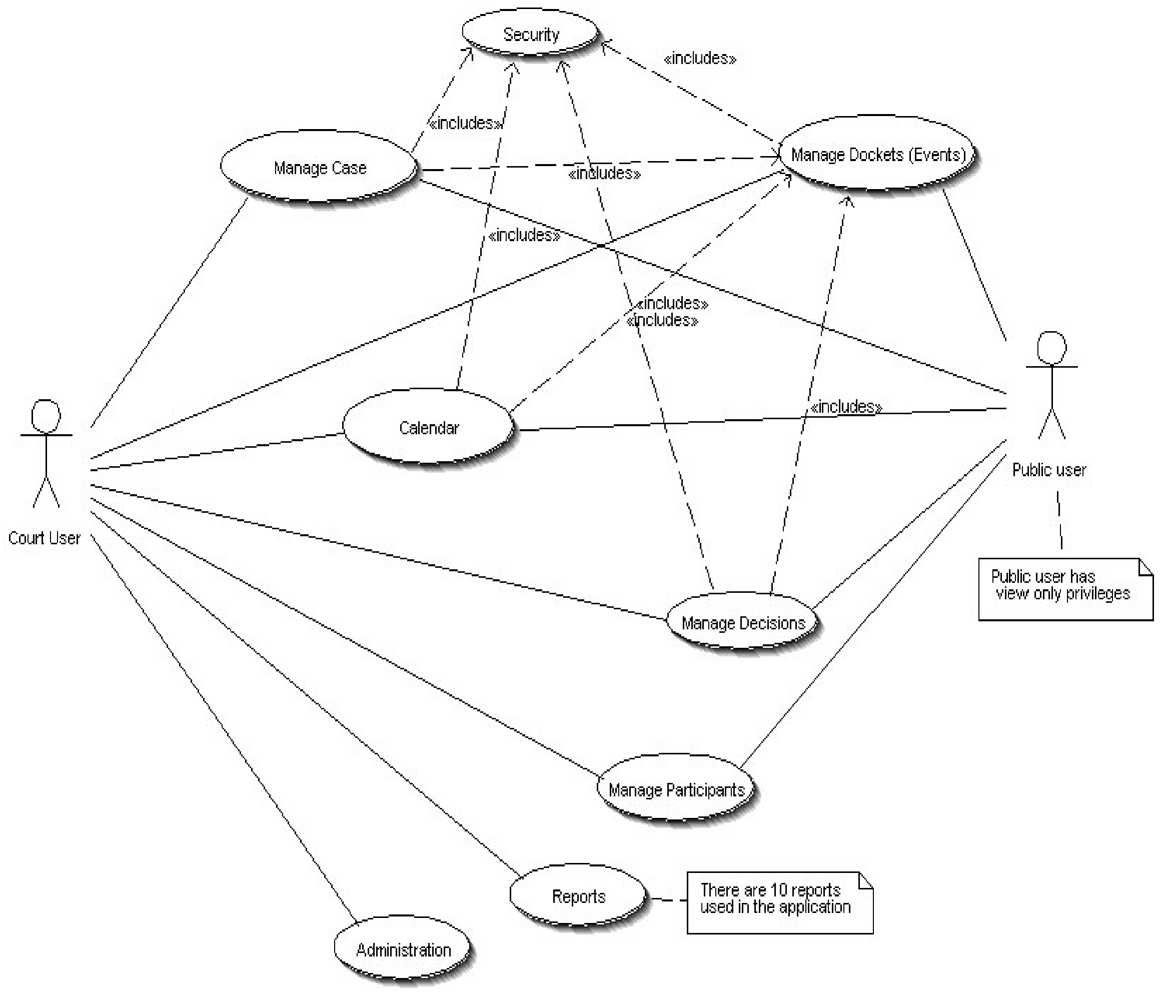
The same number of features (full & incremental) on the JIS NG architecture would be expected to consist of a total of 14,850 days, including development, test, and project management. This does not account for the significant possibility of code reuse in the JIS NG service model.

Considering the continued life span and maintenance costs associated with these applications, the savings in maintenance efforts combined with the strategic potential for further system integration and code use affords the migration to JIS NG the opportunity for significant cost savings.

APPENDIX A: ACCORDS ARCHITECTURE EVALUATION

Application Functionality

Overview of ACORDS functionality is shown in the use case diagram below



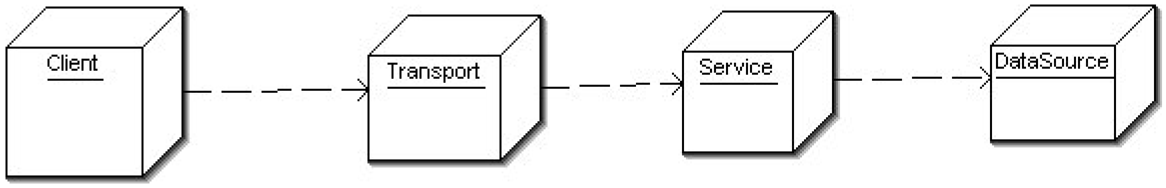


Fig 1. High Level Architecture

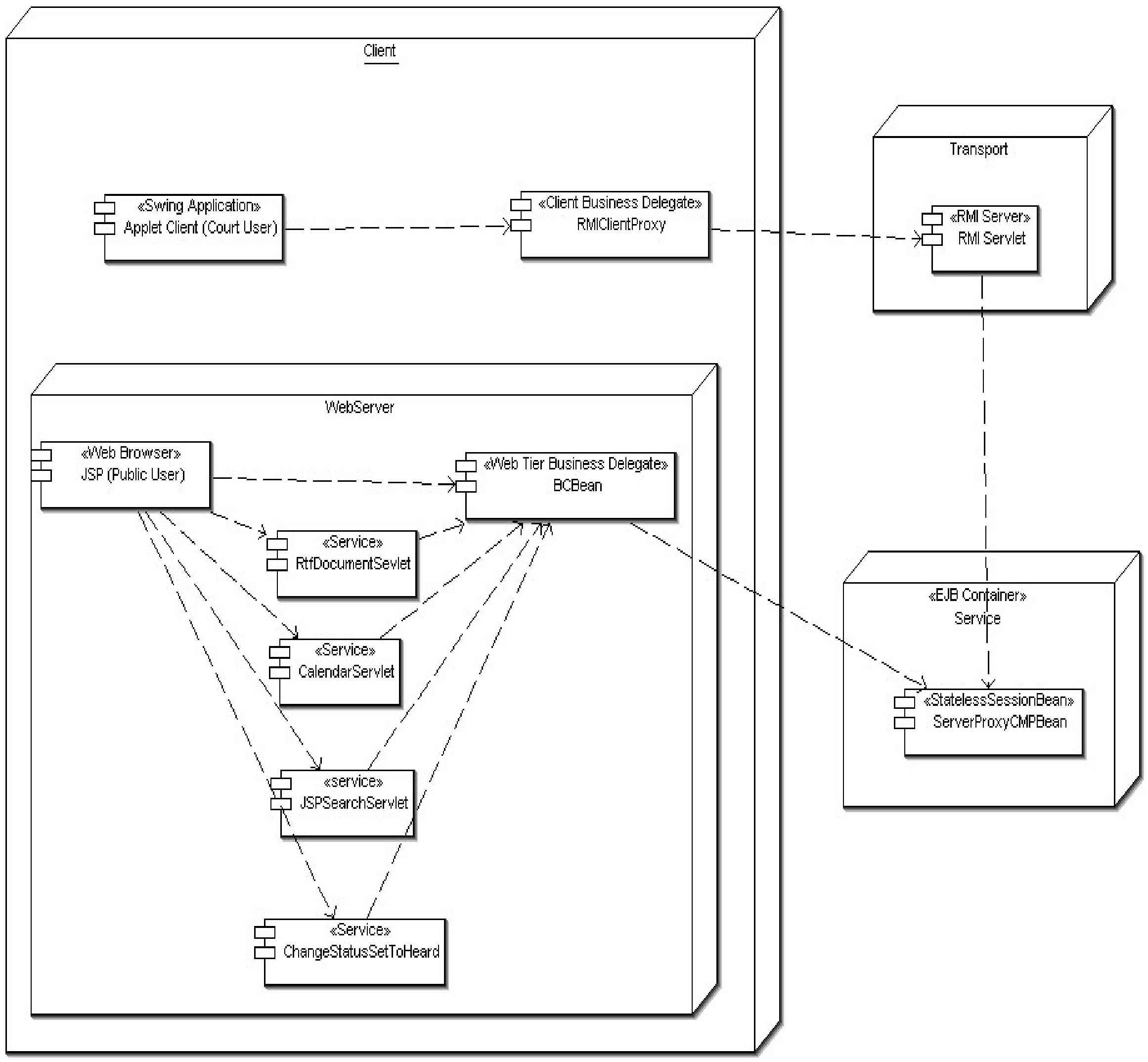


Fig 2. Client and Transport Layer

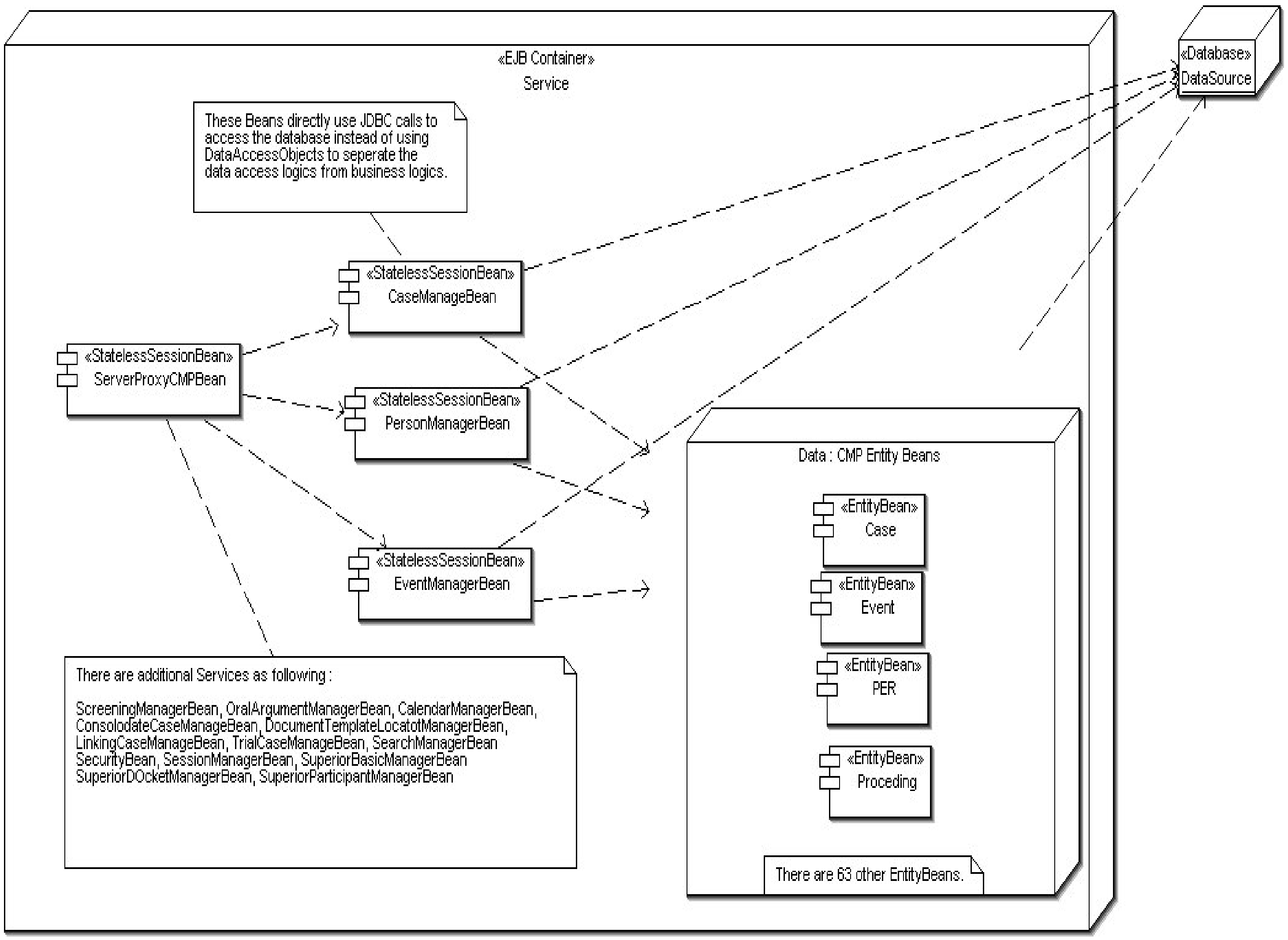

«EJB Container»
Service
These Beans directly use JDBC calls to access the database instead of using DataAccessObjects to seperate the data access logics from business logics.
«StatelessSessionBean» CaseManageBean
«StatelessSessionBean» ServerProxyCMPBean
«StatelessSessionBean» PersonManagerBean
«StatelessSessionBean» EventManagerBean
There are additional Services as following :
ScreeningManagerBean, OralArgumentManagerBean, CalendarManagerBean, ConsolodateCaseManageBean, DocumentTemplateLocatotManagerBean, LinkingCaseManageBean, TrialCaseManageBean, SearchManagerBean SecurityBean, SessionManagerBean, SuperiorBasicManagerBean SuperiorDOcketManagerBean, SuperiorParticipantManagerBean
Data : CMP Entity Beans
«EntityBean» Case
«EntityBean» Event
«EntityBean» PER
«EntityBean» Proceding
There are 63 other EntityBeans.
«Database» DataSource

Fig 3. Service and Persistence Layer

Issues with the current server side architecture

RMI Layer:
There is an additional RMI layer called the RMIServlet which serves as a transport listener in the server VM for the applet calls and delegates the calls to the session façade EJB (ServerProxyCMP). Possible reason why this layer is present is that, at the time this project started JDK might not have the classes for performing JNDI lookups. This layer may produce additional performance overhead due to the serialization/deserialization, increased complexity and development time. Taking this layer out will improve performance and maintainability.

NoDAO's:
Current architecture uses EJB CMPs for inserts and updates, and uses direct JDBC calls for retrievals. This tightly couples business logics with the persistent logics. JDBC and CMP calls should be moved to a different layer called DAO's (Data Access Objects). This separation will reduce the coupling the enterprise beans have with the persistent layer, thereby leveraging the application to have the flexibility to move to a different persistent layer like JDO, Hibernate etc.
For example there is a method call named insertCase() in the CaseManagerBean which contains more than thousand lines of code making several calls to the persistent layer without any intermediate classes.

EJB Remote Interface:
The current implementation of ACORDS uses EJB 1.1 version which provides only remote interfaces. These remote interfaces produce big performance overhead because of the serialization and de-serialization. This can be resolved by migrating the application to use EJB 2.0 (which provides local interfaces). There are currently **67 EntityBeans** and **19 SessionBeans**.

1) **ServerProxyCMPBean**:
   This layer provides only the delegation responsibility to redirect the method call from the RMIServlet to each ManagerStatelessSessionBean like EventManagerBean. This is an unnecessary and expensive layer which causes another performance and maintenance overhead. For example, currently to create a new service, following steps are required:
   a. Change Remote Interface, Home Interface, and Bean Implementation for the StatelessManagerBean.
   b. Change Remote Interface, Home Interface, and Bean Implementation for the ServerProxyCMPBean.
   c. Change Remote Interface, RMI Implementation of the RMIServlet.
   d. Change RMIClientProxy to call the new service implemented in RMIServlet.

We can cut the steps b and c by eliminating ServerProxyCMPBean, and RMIServlet, and having RMIClientProxy directly performing the lookup for the ManagerStatelessBean like EventManagerBean.

EJB Finder Methods:
The current ACORDS implementation uses vendor proprietary EJB Finder methods which make it difficult to be ported to other J2EE servers.
It is recommended to migrate to EJB 2.0 or higher, and use EJB QL queries in the deployment descriptor of the entity bean. This will help the application not to be tied to a specific type of data store.

Refactoring ManagerBean Classes:
Currently Stateless ManagerBean contains all business and data logic in just one method call. For example, CaseManagerBean contains insertCase remote method that spans more than 1000 line of codes. Some serious re-factoring has to be done here to make the application maintainable and reusable. The total line of **EventManagerBean is more than 6500** and LOC for **PersonManagerBean is 9500**. Serious re-factoring is recommended

Hard-coded business logic.
Currently there are lots of hard coded values used in repetitive conditional checks throughout the application. If this application needs to be enhanced to support other courts like superior courts, it will be extremely difficult to implement and maintain. Serious re-factoring is recommended.

For example following lines of code are taken from the CaseManagerBean and the highlighted lines show the hard coded court initials

```
if ((cmpCaseData.isTransferedCase()
|| cmpCaseData.isIsTransferredToSupremeCourt())
&& (currentCourt != null && currentCourt.equals("A08"))
&& ((oldCourt != null && oldCourt.equals("A01"))
|| (oldCourt != null && oldCourt.equals("A02"))
|| (oldCourt != null && oldCourt.equals("A03")))) {
CaseID caseNum = cmpCaseData.getOldCaseID();
String caseID = null;
if (caseNum != null)
caseID = caseNum.toJustifiedString();
String resCode = getResolutionCode(caseID, oldCourt);
if (resCode != null) {
resCode = resCode.trim();
if (resCode.equals("CERT")) {
sourceCode = "COAC";
} else {
SourceCode = "COA";
}
}
```

}

No business domain object.
Currently ACORDS uses same value objects on the server and client. This makes the application very vulnerable when a value object is changed in the server which is not needed by the client, or vice versa

Inconsistent data validation in server.
Even though there is a validation framework, in many cases developers create their own way to validate data which makes this application very hard to maintain.
For example, followings validation check appears repetitively wherever isFilingTypeValid() is called. This kind of validation should be centralized and should be done in only one place.

```
if (eventData != null
&& eventData.getFilingClass() != null
&& eventData.getFilingType() != null
&& eventData.getActionDate() != null) {
isFilingTypeValid(conn, eventData);
}
```

Inefficient Programming
There are many inefficient programming practices which make this application very difficult to maintain.

For example:

```
if (eventData instanceof BriefData) {
DataPurifierFactory
.createPurifier(BriefData.class, DataPurifierFactory.INSERT_OPERATION)
.isValid(eventData);
} else
if (eventData instanceof OpinionData)
DataPurifierFactory
.createPurifier(OpinionData.class, DataPurifierFactory.INSERT_OPERATION)
.isValid(eventData);
else
if (eventData instanceof DecisionData)
DataPurifierFactory
.createPurifier(DecisionData.class, DataPurifierFactory.INSERT_OPERATION)
.isValid(eventData);
else
  ……………………………
```

Above snippet can be replaced with one line as

DataPurifierFactory.createPurifier(eventData.getClass(),DataPurifierFactory.INSERT_OPERATION).isValid(eventData);

Overly normalized database
Since the current database schema has been too normalized, the application might suffer serious performance and maintenance problems.

For example to retrieve dockets for a case application a developer will have to access 11 tables namely **EVN, EVD, CMT, EDC, CSG, OPD, OPI, EVP, PEL, PER, PAA**.
For example EVN and EVD have one to one dependency and can be combined to one table.

Letter/Notice Generation
Currently, application uses XML templates for generating letters/documents. It then uses this template and fills in the values to create another XML which is given as input to the "XML2RTF" converter that generates a MS-WORD document. This process is pretty cumbersome, error prone and requires high maintenance. Some ways have to be though for refactoring such as Coocoon, XSLT, or etc.

Display Calendar
Current implementation of Calendar generation is error prone and requires lot of maintenance. Currently, it contains a lot of string manipulation techniques to produce well formatted XML output to take care of some special characters not recognized by XML2RTFGenerator

```
private String convertSpecialChars(String xml) {
if(xml != null) {
StringBuffer sb = new StringBuffer(xml);
for(int i=0; i < sb.length(); i++) {
char c = sb.charAt(i);
if(c == '&')
sb.setCharAt(i, ' ');
if(c == '{')
sb.setCharAt(i, '(');
if(c == '}')
sb.setCharAt(i, ')');
return sb.toString();
}
else
return xml;
}
```

From the above example it can be seen that application is converting characters like ‘&’, ‘{‘,, ‘}’ etc to spaces instead of making the XML well formatted using some standard API implementations

Self referencing joins
In the application there are numerous SQL queries performing self joins. This is a very costly database operation that can consume tremendous database resources and can affect the performance considerably.

For example:

public static final String GET_CONS_OR_LINKED_CASEIDS = "SELECT CSA_NU consCase FROM CSA,.**RLX T1, RLX T2**, RLN "
WHERE T1.RLX_TBL_TK = ? AND T1.RLX_TBL_NM = 'CSA' AND T1.RLX_RLN_TK=RLN_TK AND RLN_TYP_CD = ? AND T2.RLX_RLN_TK = RLN_TK "
AND T2.RLX_TBL_TK = CSA_TK and csa_crt_itl_nu=?";
This query performs a self join on the RLX table. Alternative way is application can execute separate SQL’s rather than performing a single self join. Also, currently it is been seen that the relationships are maintained in 2 tables namely RLN and RLX which complicates the situation even more.

Unit tests
Most of the unit tests are present in one class called ServerProxyCMPTest and contains only the basic tests for success scenarios. In short this test does not cover all boundary conditions and having this test suite run does not mean that the application is bug free.

Current client side architecture

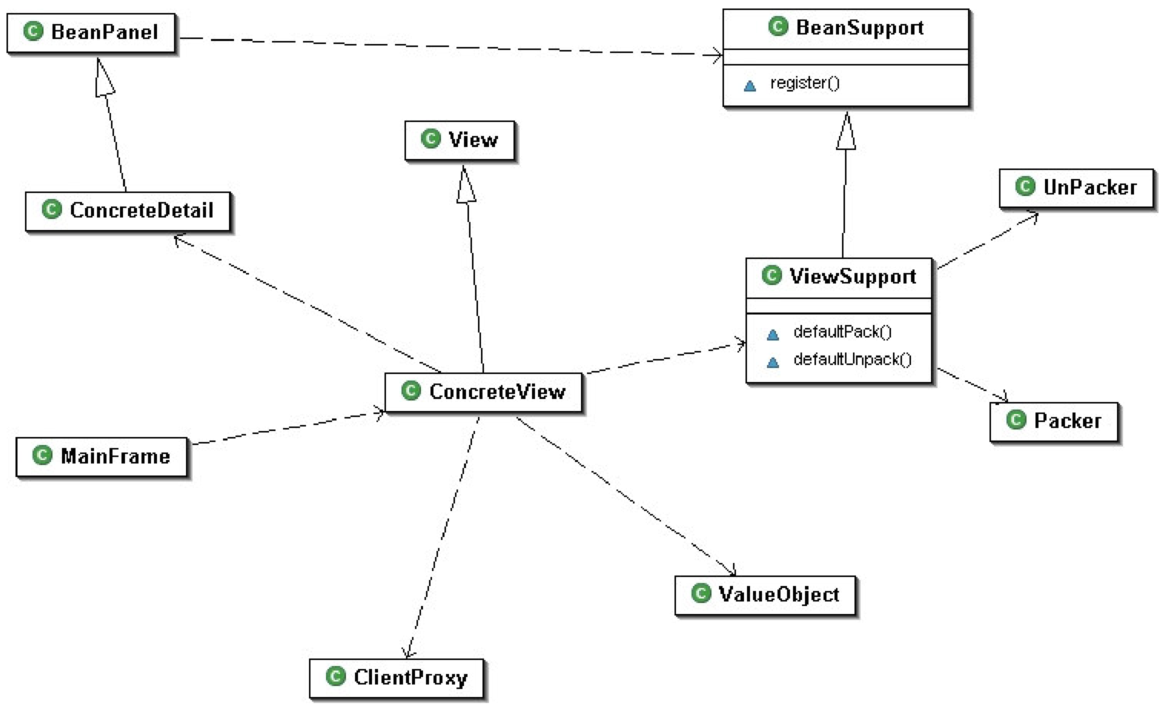


Fig 4. Client Architecture Class Diagram

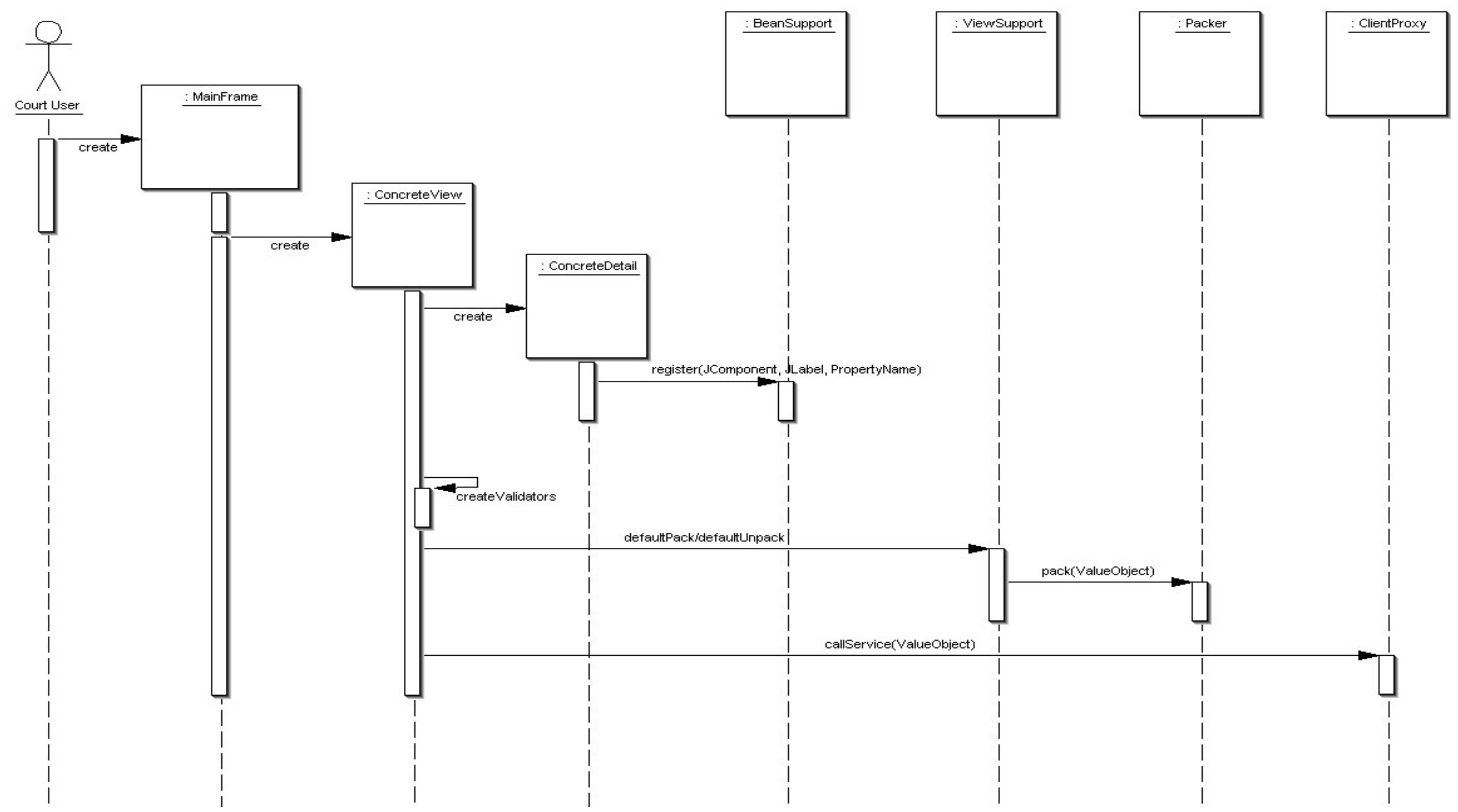


Fig 5. Client Architecture Sequence Diagram

Issues with the current client side architecture

ACORDS has multiple client applications

ACORDS has a Swing client application meant for Court users who has full update access and WEB client application meant for public users who has only view access. Maintaining two client applications requires more development time is more error prone. It is recommended for combining two client applications into a single client application with a role based security to provide a distinct access to the system.

No clear separation of View from Controller and Model.

All of the View classes contain business logics that should belong to a controller class instead. For example, ManageParticipantView contains references to view components and table models, and also coordinates the activities which a controller class is supposed to do. Since there is no clean Model-View-Controller, it is very difficult to maintain these huge View classes. For example, the lines of code for ManageParticipantsView is 1681.

Shown below are some code snippets from ManageParticipantsView that uses business logic

```
if(statusCode != null && !(statusCode.equals("A") || statusCode.equals("E") ||
statusCode.equals("F") ) ) {
...
}
```

This is clearly a business logic and should be moved to the service layer

Detail Panel

Should contain code which is generated by the tool. i.e. it should just have the layout related code for the components. This will help maintainability and also will help in isolating the area for UI designers to layout the screens.

Following code snippet is copied from ParticipantDetails and shows that it is making server calls to populate some user interface components. This is just one instance and there are numerous instances where this same case can be seen:

```
ClientProxy proxy = ClientProxy.getInstance();
proxy.setHandler(null);
roles = proxy.getParticipantRoles();
String[] countryList = proxy.getAddressCountries();
countries = new String[countryList.length+1];
countries[0] = "   ";
System.arraycopy(countryList, 0, countries, 1, countryList.length);
String[] stateList = proxy.getAddressStates();
states = new String[stateList.length+1];
states[0] = "   ";
System.arraycopy(stateList, 0, states, 1, stateList.length);
```

This code snippet should be moved to a controller class.

Client side Data Validation
The validation framework provided has been ignored and as a result, there is no central place governing the data validation and application of business rules. Instead, the individual classes hold the validation rules, which is detrimental to the development process.

```
if(pageStr != null && pageStr.trim().length() > 0) {
try {
Integer.parseInt(pageStr);
}
catch(Exception e) {
valid = false;
errorMsg = "Pages ";
}
}

if(volumeStr != null && volumeStr.trim().length() > 0) {
try {
Integer.parseInt(volumeStr);
}
catch(Exception e) {
if(!valid)
errorMsg = errorMsg + "and Volumes ";
else
errorMsg = "Volumes ";
valid = false;
}
        }

if(!valid) {
MessageDialog dialog = new MessageDialog(null,"Validation error" , true);
dialog.setMessage(errorMsg + "should be numeric");
dialog.setVisible(true);
dialog.dispose();
}
```

No Unit Tests
There is not a single unit test for the entire client side code including the framework. This has resulted in individual patches being added to the code wherever the programmers could not get satisfactory behavior from the framework. Currently the client code is reaching to a point that further maintenance is impossible unless some serious refactorings are done.

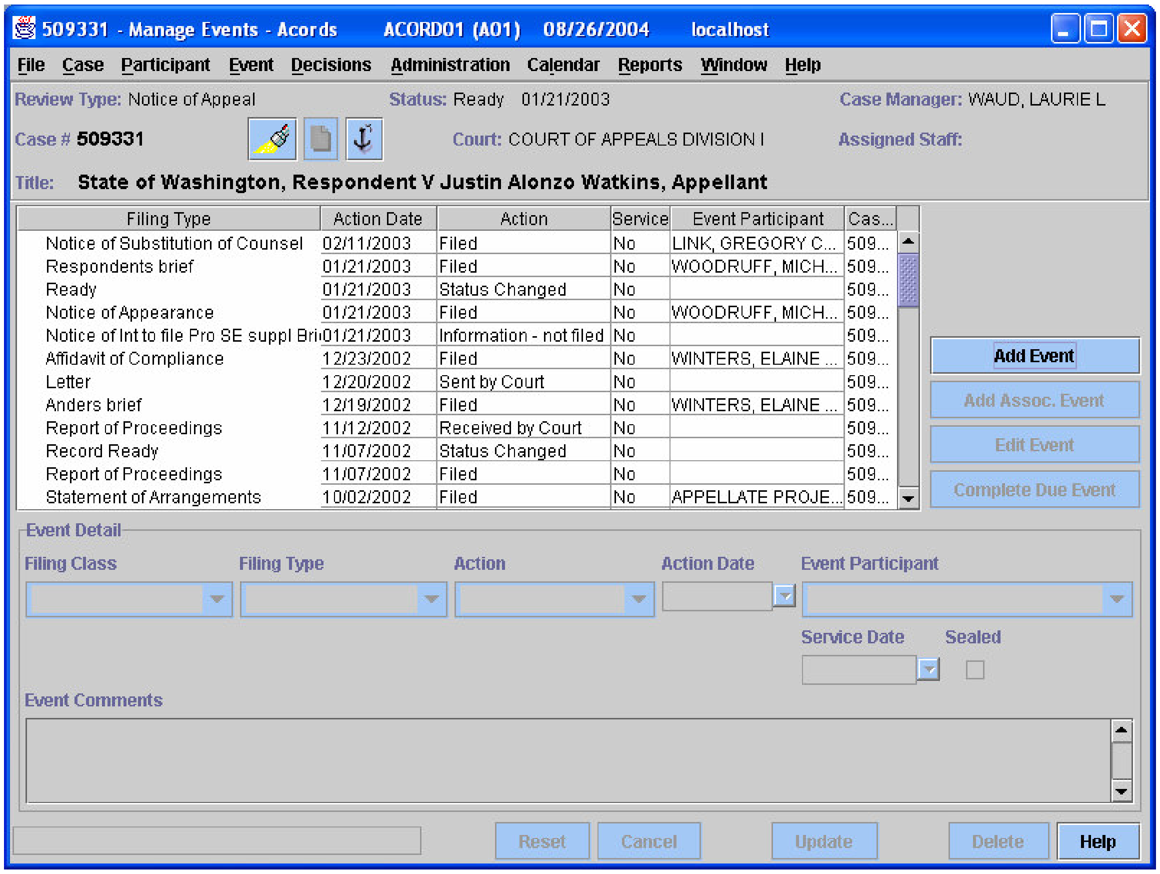
Incompatible with JDK1.4.
Screen behavior is very erratic with JDK1.4. The component layout differs a lot and so does the look and feel.

Business Logic in JSP:
Some JSP files (such as event_detail.jsp) contain significant embedded business logic. Serious re-factoring is recommend to separate business logics from JSPs. Preferably a JSP framework, like STRUTS or the one used in CAPS, should be used to enable smooth modular development.
Following code snippet is a method used in event_detail.jsp

```
private boolean isDecision(EventData eventData) {
boolean isDecision = false;
String filingClass = eventData.getFilingClass();
if(filingClass != null
        && (filingClass.trim().equals("Opinion")
        || filingClass.trim().equals("Decision") ) )
                        isDecision = true;
else if(eventData.hasSecondaryEvents()) {
                        com.netsis.acords.EventData[] secEvents =
eventData.getSecondaryEvents();
                                if(secEvents != null) {
                                        for(int i=0; i < secEvents.length; i++) {
String secFilingClass = secEvents[i].getFilingClass();
If(secFilingClass != null
        && (secFilingClass.trim().equals("Opinion")
        || secFilingClass.trim().equals("Decision") ) )
isDecision = true;
        else if(secEvents[i].hasSecondaryEvents()) {
                com.netsis.acords.EventData[] innerSecEvn =
                     secEvents[i].getSecondaryEvents();
                if(innerSecEvn != null) {
        for(int count=0; count < innerSecEvn.length; count++ ) {
                String innerSecFilingClass =
                     innerSecEvn[count].getFilingClass();
                                if(innerSecFilingClass != null
                                        && (innerSecFilingClass.trim().equals("Opinion")
                                        || innerSecFilingClass.trim().equals("Decision")
                                        ) )
        isDecision = true;
                                }
}
                                                }
                                        }
                                }
                        }
                return isDecision;
        }
```

Performance Analysis

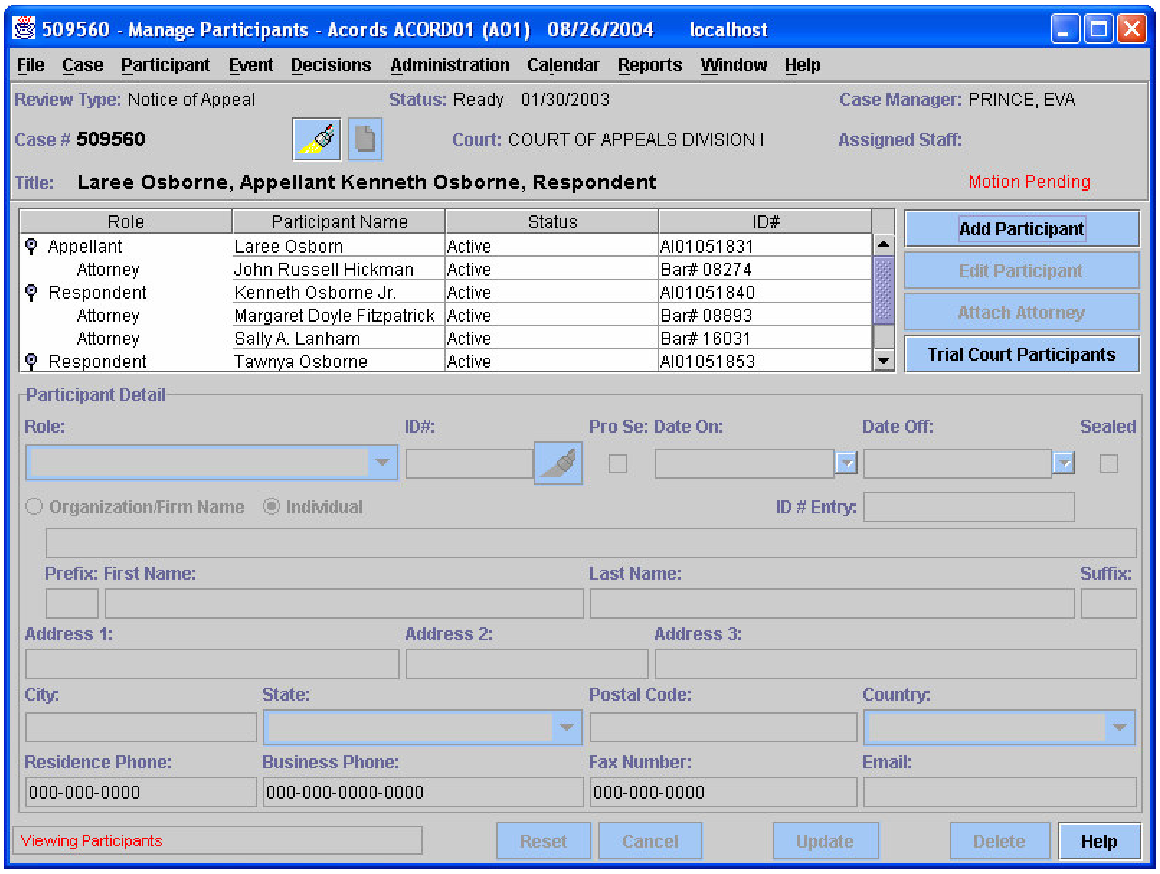
Screen name: Manage Events
Case Number: 509331
Court Initials: A01
Court Type: Appellate
Total time taken : 15.48 seconds
Total number of SQL queries executed : 149
Total time taken in the database to execute SQL queries:**10.52 seconds**
Time taken by the application: **4.96 seconds**



Review Type: Notice of Appeal Status: Ready 01/21/2003 Case Manager: WAUD, LAURIE L
Case # 509331 Court: COURT OF APPEALS DIVISION I Assigned Staff:
Title: State of Washington, Respondent V Justin Alonzo Watkins, Appellant

Filing Type	Action Date	Action	Service	Event Participant	Cas...
Notice of Substitution of Counsel	02/11/2003	Filed	No	LINK, GREGORY C...	509...
Respondents brief	01/21/2003	Filed	No	WOODRUFF, MICH...	509...
Ready	01/21/2003	Status Changed	No		509...
Notice of Appearance	01/21/2003	Filed	No	WOODRUFF, MICH...	509...
Notice of Int to file Pro SE suppl Bri	01/21/2003	Information - not filed	No		509...
Affidavit of Compliance	12/23/2002	Filed	No	WINTERS, ELAINE ...	509...
Letter	12/20/2002	Sent by Court	No		509...
Anders brief	12/19/2002	Filed	No	WINTERS, ELAINE ...	509...
Report of Proceedings	11/12/2002	Received by Court	No		509...
Record Ready	11/07/2002	Status Changed	No		509...
Report of Proceedings	11/07/2002	Filed	No		509...
Statement of Arrangements	10/02/2002	Filed	No	APPELLATE PROJE...	509...

Screen name: Manage Participants
Case Number: 509560
Court Initials: A01
Court Type: Appellate
Total time taken : 8.68 seconds
Total number of SQL queries executed : 65
Total time taken in the database to execute SQL queries: 4.98 seconds
Time taken by the application: 3.7 seconds


509560 - Manage Participants - Acords ACORD01 (A01) 08/26/2004 localhost
File Case Participant Event Decisions Administration Calendar Reports Window Help
Add Participant
Edit Participant
Attach Attorney
Trial Court Participants
Reset
Cancel
Update
Delete
Help

Review Type: Notice of Appeal
Status: Ready 01/30/2003
Case Manager: PRINCE, EVA
Case # 509560
Court: COURT OF APPEALS DIVISION I
Assigned Staff:
Title: Laree Osborne, Appellant Kenneth Osborne, Respondent
Motion Pending

Role	Participant Name	Status	ID#
Appellant	Laree Osborn	Active	AI01051831
Attorney	John Russell Hickman	Active	Bar# 08274
Respondent	Kenneth Osborne Jr.	Active	AI01051840
Attorney	Margaret Doyle Fitzpatrick	Active	Bar# 08893
Attorney	Sally A. Lanham	Active	Bar# 16031
Respondent	Tawnya Osborne	Active	AI01051853

Participant Detail
Role:
ID#:
Pro Se:
Date On:
Date Off:
Sealed
Organization/Firm Name [✓ Individual]
ID # Entry:
Prefix:
First Name:
Last Name:
Suffix:
Address 1:
Address 2:
Address 3:
City:
State:
Postal Code:
Country:
Residence Phone: 000-000-0000
Business Phone: 000-000-0000-0000
Fax Number: 000-000-0000
Email:
Viewing Participants

Source Lines of Code (excluding comments and white spaces)

Class Name	Total Lines
PersonManagerBean.java	9051
EventManagerBean.java	6506
ServerProxyCMPBean.java	6094
ServerProxyCMPTest.java	4009
CaseManagerBean.java	3908
RMIServlet.java	2902
RMIClientProxy.java	2710
MainFrame.java	1789
SecurityBean.java	1761
OralArgumentManagerBean.java	1740
ManageParticipantsView.java	1681
ScreeningManagerBean.java	1416
EventDetail.java	1342
TrialCaseManagerBean.java	1290
CaseView.java	1286
CalendarManagerBean.java	1243
ManageEventsView.java	1162
SupremeManageOpinionsView.java	1116
AppellateCaseDetail.java	1073
ConsolidateCaseManagerBean.java	1061
EventCodeServlet.java	985
ManageOpinionsView.java	965
TrackAdminView.java	905
HomeHelper.java	831
TrackManagerBean.java	814
XmlRtfConverter.java	806
DocumentGeneratorBean.java	804
OralArgumentCalenderView.java	772
SupremeOralArgumentCalenderView.java	761
SearchManagerBean.java	731
LinkingCaseManagerBean.java	651
CourtBean.java	628
UserAdministrationView.java	605
RecusalView.java	592
ConsolidateCases.java	587
SupremeOpinionDetail.java	579
ParticipantDetail.java	576
BCBean.java	570
SearchView.java	543

SuperiorParticipantManagerBean.java	494
AttorneyAdministrationView.java	479
AcordsServerUtil.java	467
EventsInfoBean.java	446
OpinionDetail.java	440
DynamicMapper.java	435
CourtOfficialMaintenanceView.java	430
NameAddressFileInfo.java	429
CourtOfficialDetail.java	423
StaticDataHelper.java	423
FinalizeCalendarView.java	416
CourtLocationMaintenanceView.java	415
OralArgumentDetail.java	412
LinkCases.java	411
FilingTypeWithShortDescHolder.java	400
PagePrinter.java	399
Header.java	398
CalendarXmlGenerator.java	393
SuperiorBasicManagerBean.java	387
HeardCasesView.java	385
DocumentTemplateLocatorView.java	376
ReassignCaseView.java	375
ScreeningInformationView.java	374
AttorneyDetail.java	365
TokenGenerator.java	362
ResourcesAdministrationView.java	359
DatePopupPanel.java	353
SecurityAdministrationView.java	351
IncompleteCaseView.java	338
ProfileAdministrationView.java	332
PendingOpinionReportView.java	330
CaseBean.java	326
RecusalViewA08.java	325
CalendarWidget.java	320
SupremeOralArgumentDetail.java	301
StaticDataHolder.java	298
View.java	292
DocumentWriter.java	292
TrackDetails.java	275
PerfectionView.java	274
CaseNumberFormatter.java	271
AccessControlList.java	259
Filer.java	258

CommonDate.java	255
OverdueEventsView.java	254
PendingClosureView.java	253
MilestoneDetails.java	249
ViewSupport.java	249
PRPStatusView.java	247
ProfilePermissionAdministrationView.java	245
TableSorter.java	244
AppellateCaseData.java	243
CasesToSupremeCourtView.java	240
ServerProxyCMPTestConstants.java	240
Resources.java	239
AbstractTestCase.java	239
AssignmentJudge.java	236
EventBean.java	235
ScreeningTrialCourt.java	234
ClientProxy.java	232
ScreenBean.java	231
CasesWithoutOpinionView.java	230
PendingWithoutDueView.java	230
ServerProxyCMP.java	230
CaseTitle.java	224
ManageEventsResultData.java	224
ScreenedCasesView.java	223
DocumentBean.java	220
TransferFrom.java	210
RMIProxy.java	210
AcordsDataPurifier.java	210
TrialCourt.java	209
ReadyCaseView.java	204
DocumentTemplateLocatorManagerBean.java	202
CalendarServlet.java	201
CaseBanner.java	200
LocationDetail.java	192
ProceedingBean.java	192
FilingActionHolder.java	191
DisciplinaryHearing.java	189
UserDetail.java	185
DataHolder.java	184
SessionServerConstants.java	183
ValidatorTester.java	178
PAABean.java	177
Replace.java	174

PersonFinder.java	174
P.java	173
EvcEvaUpdate.java	171
DACAttorneyPopup.java	166
DefaultPropertyHandler.java	166
TransferCaseView.java	158
ChangeStatusSetToHeard.java	158
TrackBean.java	157
PELBean.java	157
OpinionListView.java	156
ServerConstants.java	156
CaseIDCacheHandler.java	155
DocumentTemplateLocatorDetail.java	154
ParticipantData.java	154
ActionSelect.java	154
FilingTypeHolder.java	153
AttorneySearcher.java	152
BeanSupport.java	151
Copy.java	148
RtfDocumentServlet.java	147
SuperFileReaderImpl.java	147
AclGenerator.java	146
AcordsPropertyPurifierFactory.java	146
ChangeTracksView.java	141
RPABean.java	140
SuperiorBasicData.java	136
ADBean.java	136
MilestoneBean.java	134
OFLBean.java	134
AcordsDatabaseMgmr.java	132
CodeReaderImpl.java	132
DataValidator.java	132
PublicCalendarData.java	131
DataTypeConvertor.java	131
SuperiorSentenceData.java	131
ChangeTracksDetails.java	130
PASBean.java	129
SuperiorDocketManagerBean.java	128
BasicExceptionHandler.java	126
DTEBean.java	126
FilingClassHolder.java	125
SecurityTypeHolder.java	125
FilingDateBean.java	125

EvnDateBean.java	124
ADABean.java	124
ReviewTypeHolder.java	121
USRBean.java	121
JTreeTable.java	119
ScreeningCaseData.java	118
GridPanel.java	117
OacDateFormatter.java	116
PNDBean.java	114
PPABean.java	114
SecurityBottomDetail.java	112
SuperiorChargeData.java	111
OCOBean.java	110
JNDIHelper.java	109
Delete.java	109
SecurityManagerTest.java	109
ScreeningScreener.java	108
StateCountryConverter.java	108
PADBean.java	108
AppellateAssignCaseDialog.java	107
DataPropertyInspector.java	107
SessionManagerBean.java	107
EvcEvaInsert.java	105
CategoryHolder.java	104
ErrorMessageDialog.java	104
SecurityTopDetail.java	103
CategoryBean.java	103
PERBean.java	103
OralArgumentJudgesHolder.java	102
ProceedingOfficialBean.java	101
TrialCaseData.java	100
EvcAll.java	100
SealedCaseDisplayWindow.java	99
EventData.java	99
DataPurifier.java	99
FileHelper.java	99
LogEntry.java	98
EventParticResultData.java	97
MaskDocument.java	96
SeverStatusCheckHolder.java	95
SourceCodeHolder.java	95
ProfileDetail.java	94
StatusCheckHolder.java	94

SessionTrackerData.java	94
EventCodeSelect.java	94
PropertyPurifierFactory.java	93
ProfilePermissionBottomDetail.java	92
SearchManagerBeanTest.java	92
TrialCaseBean.java	92
AOBBean.java	92
ATZBean.java	92
LawClerkHolder.java	91
JMaskField.java	91
OpiDecisionBean.java	91
PersonUtil.java	89
ScreeningBottom.java	88
CourtNameHolder.java	88
CacheLoader.java	88
FederalCourtInfo.java	87
SimpleFilenameFilter.java	87
ATPBean.java	87
RlxBean.java	86
PEMBean.java	86
CaseIssues.java	85
AppellateCourtNameHolder.java	85
StatusHistoryBean.java	85
AcceleratedReasonsHolder.java	84
ParserManager.java	84
SPLogManager.java	84
ResourcesDetail.java	83
EventCodeActionDelete.java	83
PPHBean.java	83
OralHearingLocationHolder.java	82
ValidatorFactory.java	82
ORGBean.java	82
CacheSQLHelper.java	81
ValidationTester.java	81
UnqualifiedConnectionDelegate.java	81
PEABean.java	81
CaseComplexity.java	80
CalendarPopupButton.java	80
SecurityContext.java	79
DbConnManager.java	79
PSABean.java	79
EventsSQLHelper.java	78
AuthenticationMgmrBean.java	78

ContextualAclEntry.java	77
ProceedingHistoryBean.java	77
CaseStatusSequenceHolder.java	76
DateCalculator.java	76
DateCalculator.java	76
GeographicRegionHolder.java	75
MotionOfficialsHolder.java	75
SpaBean.java	75
ArgumentAllotmentTimeHolder.java	74
CalendarTypeHolder.java	74
CaseComplexityHolder.java	74
DisabilityAppealJudgementsHolder.java	74
DisabilityPetitionJudjementHolder.java	74
DisabilityTransferToJudjementHolder.java	74
FinancialJudjementHolder.java	74
IssueHolder.java	74
MiscJudjementHolder.java	74
MotionCalendarTypeHolder.java	74
OralArgDurationHolder.java	74
OralArgTimeHolder.java	74
ScrInfoCalTypeHolder.java	74
Utility.java	74
CommentTitleBean.java	74
BasicScreeningCaseData.java	73
CaseProperties.java	73
MotionsHearingLocationHolder.java	73
MotionStatusHolder.java	73
ReadOnlyEventsHolder.java	73
StateValidator.java	73
AppellateCaseSearchDialog.java	72
JudgeNameHolder.java	71
ScreenerNameHolder.java	71
SupremeCaseData.java	71
SteppedComboBoxUI.java	71
ProceedingOfficialHistoryBean.java	71
LinkingSeverReasonsHolder.java	70
NonDisciplinelCaseTypeHolder.java	70
SeverReasonsHolder.java	70
SubSystemHolder.java	70
Search.java	70
ORABean.java	70
PhaBean.java	70
MilestoneData.java	68

SimpleTreeTableModel.java	68
AppellateCaseAssigneeDialog.java	67
CaseStatusHolder.java	67
ConsolidatedReasonsHolder.java	67
CourtNameInitialsHolder.java	67
FilingFeeHolder.java	67
LinkingReasonsHolder.java	67
MilestoneDateTypeHolder.java	67
RoleHolder.java	67
RoleOfficialHolder.java	67
IconLabel.java	67
AddressData.java	66
AppellateCaseTypeHolder.java	66
AttorneyAdmissionJudgementHolder.java	66
AttorneyAdmissionsCaseTypeHolder.java	66
AttorneyCategoriesHolder.java	66
AttorneyStatusHolder.java	66
AttorneySubTypeHolder.java	66
CountryHolder.java	66
DisciplinaryActionCaseTypeHolder.java	66
DisciplinaryHearingJudgementHolder.java	66
JudgementHolder.java	66
JudicialAppealCaseTypeHolder.java	66
JudicialAppealJudgementHolder.java	66
OfficialSubTypesHolder.java	66
OfficialTypesHolder.java	66
StateHolder.java	66
TrialCaseTypeHolder.java	66
JSPSearchServlet.java	66
ProfilePermissionTopDetail.java	65
Builder.java	65
MilestoneHolder.java	65
JComponentCellEditor.java	65
OpinionBean.java	65
PropertyChangeEventSupport.java	64
RlnBean.java	64
HistoryBean.java	64
OpinionData.java	63
HolidaysHolder.java	63
OANBean.java	63
EvnParticipantBean.java	63
ScreeningComments.java	62
OralArgumentScheduleData.java	62

PersonManager.java	62
Searcher.java	61
CaseResolutionHolder.java	61
MotionDecisionHolder.java	61
OpinionDecisionHolder.java	61
OpinionTypeHolder.java	61
PanelRoleHolder.java	61
PublishingStatusHolder.java	61
FileEditor.java	61
MotionData.java	60
ConsolidatedCodeHolder.java	60
ResolutionReasonHolder.java	60
TreeTableModelAdapter.java	60
RemoteHttpServlet.java	60
OOTBean.java	60
PHNBean.java	60
CTCBean.java	60
OpinionNumberGenerator.java	59
NoticeProceedingData.java	59
PropertyChangeEventQueueItem.java	59
DbSQLHelper.java	58
AclPolicy.java	58
MaskParser.java	58
DefaultPropertyPacker.java	58
ConnectionCode.java	58
JSPTableModel.java	58
PermissionCollection.java	57
ReassignCasesTableRenderer.java	57
DocumentTemplateLocatorData.java	57
ParticipantNonProSePurifier.java	57
MessageDialog.java	57
TrialCaseInfoBean.java	57
MotionBean.java	57
CTIData.java	56
RecusalControls.java	55
SearchCaseData.java	55
DateSelectionDialog.java	55
CopBean.java	55
CourtLocationData.java	54
EVDTableData.java	54
OverDueEventDescriptionData.java	54
SearchRACF.java	54
PropertyPurifier.java	54

SortableTableModel.java	54
TrackHistoryBean.java	54
Login.java	53
SealedCaseCellRenderer.java	53
ExceptionHandler.java	53
TrackData.java	53
PropertyIteratorImpl.java	53
EventManager.java	53
AccessControlEntry.java	52
CourtOfficialData.java	52
CalendarSQLHelper.java	52
PurificationException.java	52
StringUtil.java	52
FriendlyTextArea.java	52
EEIconTable.java	51
Purge.java	51
ATYBean.java	51
ClientUtil.java	50
NumberFormatter.java	50
CompositeValidator.java	50
TrialCaseFillingDateBean.java	50
SenBean.java	50
SearchEmployee.java	49
PropertyChangeEventQueue.java	49
PCMBean.java	49
CalendarUtil.java	48
SearchField.java	48
Clock.java	47
ScreeningViewData.java	47
OpinionNumberData.java	47
EvnSelect.java	47
ArrayDataPurifier.java	46
DateComparisonValidator.java	46
ConnectionInfo.java	46
CaseManagerSQLHelper.java	45
BackgroundLoader.java	45
SuperiorDocketData.java	45
EMABean.java	45
EventProperties.java	44
PanelMemberData.java	44
PhoneData.java	44
NoticeEventData.java	44
CourtSQLHelper.java	44

HttpMessage.java	44
ScjBean.java	44
EventParticipantData.java	43
UserData.java	43
DefaultPropertyUnpacker.java	43
DTEKey.java	43
SpaKey.java	43
CompressCharacters.java	43
RowProperties.java	43
ScreeningStatus.java	42
ADData.java	42
DataFormatter.java	42
IndexManager.java	41
Env.java	41
MaskTokenizer.java	41
SuperiorParticipantData.java	41
CategoryKey.java	41
OpiParticipantBean.java	41
RPAKey.java	41
ChangeTracksDetailsBeanInfo.java	40
ProfileDetailBeanInfo.java	40
ProfilePermissionBottomDetailBeanInfo.java	40
ProfilePermissionTopDetailBeanInfo.java	40
ResourcesDetailBeanInfo.java	40
SecurityBottomDetailBeanInfo.java	40
SecurityTopDetailBeanInfo.java	40
UserDetailBeanInfo.java	40
CompositePurifier.java	40
JTextFieldFilter.java	40
PSAKey.java	40
CaseAssigneeData.java	39
CourtAssignedStaffData.java	39
IndividualData.java	39
NoticeCaseData.java	39
DataPurifierFactory.java	39
CopyPastePopupMenu.java	39
EventQueueWatchdog.java	39
Rename.java	39
CommentTitleKey.java	39
SFCBean.java	39
MyColorRenderer.java	38
EventsViewData.java	38
PublicCalendarViewData.java	38

JspErrorHandler.java	38
CurrentOS390Time.java	38
DataTypeConvertor.java	38
EMLBean.java	38
Applet.java	37
MaskMacros.java	37
Debug.java	37
AbstractCellEditor.java	37
SSNValidator.java	37
EvnParticipantKey.java	37
OralArgumentPendingData.java	36
ParticipantProperties.java	36
FriendlyTextField.java	36
NameFormatter.java	36
CodeBean.java	36
PNDKey.java	36
CaseViewBeanInfo.java	35
PendingOpinionReportViewBeanInfo.java	35
ViewPanel.java	35
Trace.java	35
EventOpinionBean.java	35
ORAKey.java	35
PEAKey.java	35
ATZKey.java	35
AclManager.java	34
PELData.java	34
SPExecutorBean.java	34
ShortEventData.java	34
Case.java	34
TrialCaseFillingDateKey.java	34
SenKey.java	34
ActionCodeKey.java	34
PAA.java	34
BasicAppellateCaseData.java	33
DocumentTemplateLocatorBeanInfo.java	33
ScreeningInformationViewBeanInfo.java	33
XmlTagConstants.java	33
ValidatorSpecifier.java	33
SQLStatements.java	33
OCOKey.java	33
ProceedingOfficialHistoryKey.java	33
TrackHistoryKey.java	33
SocialSecurityNumberFormatter.java	33

DocumentKey.java	33
StatusHistoryKey.java	33
PADKey.java	33
PASKey.java	33
PEMKey.java	33
PPAKey.java	33
PPHKey.java	33
AttorneyData.java	32
FilingTypeData.java	32
OralArgCalendarViewData.java	32
SecuritySQLHelper.java	32
UserProfileData.java	32
ActionCodeBean.java	32
CaseID.java	31
MainFrameBeanInfo.java	31
OralArgumentCalenderViewBeanInfo.java	31
SupremeOralArgumentCalenderViewBeanInfo.java	31
ScreeningInformationProperties.java	31
TableMap.java	31
ProceedingOfficialKey.java	31
ProceedingHistoryKey.java	31
ScreenKey.java	31
FilingDateKey.java	31
MilestoneKey.java	31
RlnKey.java	31
RlxKey.java	31
TrialCaseInfoKey.java	31
ScjKey.java	31
EvnDateKey.java	31
HistoryKey.java	31
EventKey.java	31
EventOpinionKey.java	31
MotionKey.java	31
OpiParticipantKey.java	31
ADAKey.java	31
EMAKey.java	31
ORGKey.java	31
PAAKey.java	31
PCMKey.java	31
PELKey.java	31
PERKey.java	31
PhaKey.java	31
BasicCalendarDataImpl.java	30

PublicCalendarParticipantData.java	30
ParticipantSQLHelper.java	30
CenteringDialog.java	30
SuperiorDocketTextData.java	30
CacheUpdateServlet.java	30
ConnectionProperties.java	30
TrialCourtBeanInfo.java	29
CaseAssigneeRenderer.java	29
RecusalViewA08BeanInfo.java	29
RecusalViewBeanInfo.java	29
AdditionalCaseData.java	29
PAAData.java	29
PERData.java	29
EnvRACF.java	29
BeanPanel.java	29
ProceedingKey.java	29
OANKey.java	29
OOTKey.java	29
TrackKey.java	29
OpiDecisionKey.java	29
OpinionKey.java	29
ADKey.java	29
ATYKey.java	29
EMLKey.java	29
OFLKey.java	29
PAS.java	29
PHNKey.java	29
USRKey.java	29
SecurityManager.java	29
CaseComments.java	28
IncompleteCaseViewBeanInfo.java	28
OralArgCalendarSQLHelper.java	28
ExceptionCellRenderer.java	28
DateCompare.java	28
PEL.java	28
PrincipalGroup.java	27
SecurityHandler.java	27
EventDetailBeanInfo.java	27
PRPStatusViewBeanInfo.java	27
ReadyCaseViewBeanInfo.java	27
UserAdministrationViewBeanInfo.java	27
ProfileData.java	27
RecusalData.java	27

CustomRMISocketFactory.java	27
DynamicMapperSQLHelper.java	27
OriginalActionPurifier.java	27
SuperiorDefendantData.java	27
Proceeding.java	27
FinalizeCalenderViewBeanInfo.java	26
HeardCasesViewBeanInfo.java	26
OverdueEventsViewBeanInfo.java	26
PendingClosureViewBeanInfo.java	26
ScreenedCasesViewBeanInfo.java	26
Data.java	26
HolidaysData.java	26
ReviewTypeData.java	26
CalendarServerConstants.java	26
DuplicateHashTable.java	26
Mkdir.java	26
Track.java	26
Event.java	26
ChangeTracksViewBeanInfo.java	25
PerfectionViewBeanInfo.java	25
ProfileAdministrationViewBeanInfo.java	25
ProfilePermissionAdministrationViewBeanInfo.java	25
ResourcesAdministrationViewBeanInfo.java	25
SecurityAdministrationViewBeanInfo.java	25
SearchSQLHelper.java	25
FormattedTextArea.java	25
URLDecoder.java	25
EventCodeData.java	25
Court.java	25
ADA.java	25
PAD.java	25
CourtLocationMaintenanceViewBeanInfo.java	24
ManageOpinionsViewBeanInfo.java	24
OpinionListViewBeanInfo.java	24
SupremeManageOpinionsViewBeanInfo.java	24
DecisionData.java	24
FilerDetailData.java	24
OpinionDescriptionData.java	24
UserPrincipal.java	24
AcordsUserManager.java	24
EVNData.java	24
ParticipantInfo.java	24
PHAData.java	24

PNDData.java	24
JSPMessage.java	24
CaseManager.java	24
AcordsPropertyHandler.java	23
ConsolidateCasesBeanInfo.java	23
CourtOfficialMaintenanceViewBeanInfo.java	23
LinkCasesBeanInfo.java	23
ManageEventsViewBeanInfo.java	23
TrackAdminViewBeanInfo.java	23
AcordsException.java	23
TrialCaseKey.java	23
AcordsException.java	23
PER.java	23
USR.java	23
AccessControlContext.java	22
AppellateCaseDetailBeanInfo.java	22
CaseIssuesBeanInfo.java	22
CaseTitleBeanInfo.java	22
FilerBeanInfo.java	22
RecusalControlsBeanInfo.java	22
ScreeningBottomBeanInfo.java	22
ScreeningCommentsBeanInfo.java	22
ScreeningScreenerBeanInfo.java	22
ScreeningStatusBeanInfo.java	22
ScreeningTrialCourtBeanInfo.java	22
SupremeOpinionDetailBeanInfo.java	22
TrackDetailsBeanInfo.java	22
ViewBeanInfo.java	22
AttorneyAdministrationViewBeanInfo.java	22
FederalCourtData.java	22
OfficialData.java	22
PRPCaseData.java	22
ResourceData.java	22
ResourceAccessLevel.java	22
CharacterValidator.java	22
Milestone.java	22
PAAHome.java	22
PEM.java	22
CaseComplexityBeanInfo.java	21
CourtOfficialDetailBeanInfo.java	21
MilestoneDetailsBeanInfo.java	21
OpinionDetailBeanInfo.java	21
SupremeOralArgumentDetailBeanInfo.java	21

ReassignCaseViewBeanInfo.java	21
AcordsNameFormatter.java	21
JComponentRenderer.java	21
FilingDate.java	21
SFCKey.java	21
PPA.java	21
PPH.java	21
CTCKey.java	21
AclPermission.java	20
AttorneyDetailBeanInfo.java	20
CaseBannerBeanInfo.java	20
LocationDetailBeanInfo.java	20
OralArgumentDetailBeanInfo.java	20
ParticipantDetailBeanInfo.java	20
TransferFromBeanInfo.java	20
ResourceNames.java	20
OpiDecisionData.java	20
AcordsServerDataSorter.java	20
JspSecurityUtil.java	20
PropertiesReader.java	20
PropertiesReader.java	20
StatusHistory.java	20
ORG.java	20
PEA.java	20
SecurityPrincipal.java	19
BasicOralArgCaseData.java	19
BriefData.java	19
CaseViewData.java	19
FederalCourtInfoBeanInfo.java	19
HeaderBeanInfo.java	19
EventsInfoSQLHelper.java	19
XmlToRtf.java	19
AssignerFactory.java	19
TrialCourtInfo.java	19
PropertiesReader.java	19
Debug.java	19
Document.java	19
RPA.java	19
AOB.java	19
AOBKey.java	19
AssignmentJudgeBeanInfo.java	18
DisciplinaryHearingBeanInfo.java	18
FilingClassData.java	18

SecurityConstants.java	18
CalendarMonthRenderer.java	18
QueueMapEntry.java	18
TrimLeadingZero.java	18
OralArgumentManager.java	18
DTE.java	18
OpiDecision.java	18
PSA.java	18
CaseCommentsBeanInfo.java	17
PermissionData.java	17
ValidationError.java	17
CaseKey.java	17
CodeReaderContentsImpl.java	17
OacHighLowDates.java	17
CodeKey.java	17
EvnDate.java	17
ATZ.java	17
MilestoneProperties.java	16
RecusalProperties.java	16
ResourcePermission.java	16
SecurityProperties.java	16
NoticeParticipantData.java	16
PublicCalendarCourtData.java	16
TrackSQLHelper.java	16
ResourceNameMapper.java	16
JComponentCellRenderer.java	16
ParticipantProcessControlData.java	16
ConsolidateCaseManager.java	16
OOT.java	16
Rlx.java	16
EventHome.java	16
ATP.java	16
ViewDescriptor.java	15
AcordsServerException.java	15
SQLDateAssigner.java	15
MaskCondition.java	15
CompareArray.java	15
ProceedingHome.java	15
CaseHome.java	15
CopKey.java	15
TrackHome.java	15
EventsInfo.java	15
PADHome.java	15

Pha.java	15
ATPKey.java	15
CourtData.java	14
CourtInfo.java	14
DefaultSupremeJudgeData.java	14
JudgeNameData.java	14
OralArgumentProperties.java	14
RecusalA08Properties.java	14
ParticipantInformation.java	14
TrialCaseSQLHelper.java	14
TrackProperties.java	14
MaskToken.java	14
ViewPropertyIteratorImpl.java	14
BetweenValidator.java	14
SharedPackageTest.java	14
ScreeningManager.java	14
LinkingCaseManager.java	14
TrialCaseManager.java	14
OAN.java	14
CodeReaderImpl_JDBC.java	14
RowInfo.java	14
EvnDateHome.java	14
PND.java	14
TestClient.java	13
BasicCalendarData.java	13
CaseIDData.java	13
DocumentTemplateLocatorProperties.java	13
ReadyCaseData.java	13
CustomDateAssigner.java	13
SortCriteria.java	13
MaskSet.java	13
PLAF.java	13
AbstractPacker.java	13
MultiColumnObject.java	13
DateValidator.java	13
CalendarManager.java	13
Screen.java	13
Cop.java	13
RlxHome.java	13
EditableSortableModel.java	12
ParticipantManipulation.java	12
PropertyPacker.java	12
ScreenedCasesReportData.java	12

SearchProperties.java	12
ConstantDataValues.java	12
TrackManager.java	12
OFL.java	12
PELHome.java	12
ResourceTypes.java	11
DataPropertyIteratorImpl.java	11
LimitedStyledDocument.java	11
UneditableTableModel.java	11
ProxyPackageTest.java	11
ProceedingOfficial.java	11
Opinion.java	11
DocumentGenerator.java	11
MaxLengthValidator.java	10
Sen.java	10
NoDVIParticipantException.java	10
EvnParticipant.java	10
PCM.java	10
PPAHome.java	10
PSAHome.java	10
ATZHome.java	10
SuperiorParticipantManager.java	10
SecurityManager.java	9
CaseStatus.java	9
OrganizationData.java	9
GenerateSequence.java	9
MaskExpression.java	9
MaskLiteral.java	9
ActionProxy.java	9
TreeTableModel.java	9
MinLengthValidator.java	9
SQLHelper.java	9
SQLHelper.java	9
FilingDateHome.java	9
Scj.java	9
InvalidParameterException.java	9
OpinionHome.java	9
ADHome.java	9
PEMHome.java	9
PERHome.java	9
PPHHome.java	9
RPAHome.java	9
PropertyUnpacker.java	8

ExceptionTypes.java	8
FieldValidationException.java	8
RecordNotExistException.java	8
TrimAssigner.java	8
PropertyHandler.java	8
Time.java	8
AlphabeticValidator.java	8
AlphanumericValidator.java	8
NumericValidator.java	8
ProceedingOfficialHome.java	8
ProceedingOfficialHistory.java	8
CommentTitle.java	8
CommentTitleHome.java	8
DTEHome.java	8
RlnHome.java	8
TrialCase.java	8
Spa.java	8
IncompleteInformationException.java	8
InvalidCaseTypeException.java	8
InvalidDateFormatException.java	8
InvalidLogonException.java	8
InvalidPersonTypeException.java	8
ScomisCaseOnlyException.java	8
SuperFileReader.java	8
DocumentHome.java	8
ADAHome.java	8
EMA.java	8
PASHome.java	8
PEAHome.java	8
AclContext.java	7
AclSource.java	7
ExceptionSubTypes.java	7
NotNullDataPurifier.java	7
NullDataPurifier.java	7
BasicBeanInfo.java	7
PropertyChangeSource.java	7
PropertyIterator.java	7
NotNullValidator.java	7
NullOrEmptyValidator.java	7
NullValidator.java	7
StringValidator.java	7
TrimmedValueNotEmptyValidator.java	7
Validator.java	7

Category.java	7
CategoryHome.java	7
Rln.java	7
TrialCaseHome.java	7
AuthenticationMgmr.java	7
EventOpinionHome.java	7
EvnParticipantHome.java	7
Motion.java	7
OpiDecisionHome.java	7
ORA.java	7
PhaHome.java	7
PNDHome.java	7
SuperiorBasicManager.java	7
AccessLevel.java	6
AclEntryNotFoundException.java	6
OpinionNotExistException.java	6
QueryNotExecutedException.java	6
UserAuthFailedException.java	6
Assigner.java	6
BlankAssigner.java	6
CurrentDateAssigner.java	6
EmptyPurifier.java	6
InputDataException.java	6
Purifier.java	6
ValidationException.java	6
ValidationWarning.java	6
EnterKeyListener.java	6
ValidatingBoundPropertyChangeSource.java	6
ConnectionBusyException.java	6
ConnectionCreateException.java	6
DriverNotLoadedException.java	6
PropertiesNotFoundException.java	6
PropertyNotFoundException.java	6
AfterValidator.java	6
AnyStringValidator.java	6
BeforeValidator.java	6
SQLHelper.java	6
SQLHelper.java	6
LogManager.java	6
OCO.java	6
DocumentTemplateLocatorManager.java	6
CourtHome.java	6
MilestoneHome.java	6

OANHome.java	6
TrialCaseInfo.java	6
SpaHome.java	6
AuthenticationMgmrHome.java	6
CodeReader.java	6
PropertiesNotFoundException.java	6
OpiParticipant.java	6
OpiParticipantHome.java	6
StatusHistoryHome.java	6
AD.java	6
EMAHome.java	6
OFLHome.java	6
ORAHome.java	6
PCMHome.java	6
AOBHome.java	6
ATPHome.java	6
ConsolidationStatus.java	5
FinalizeProperties.java	5
HeardProperties.java	5
ErrorCodes.java	5
AbstractUnpacker.java	5
SQLHelper.java	5
ProceedingOfficialHistoryHome.java	5
TrackHistoryHome.java	5
TrialCaseFillingDate.java	5
SFC.java	5
SFCHome.java	5
HistoryHome.java	5
MotionHome.java	5
ATY.java	5
ATYHome.java	5
EML.java	5
EMLHome.java	5
ORGHome.java	5
PHNHome.java	5
CTCHome.java	5
USRHome.java	5
SearchManager.java	5
ContextualPolicy.java	4
Policy.java	4
CalendarProcessList.java	4
SearchViewBeanInfo.java	4
TransferCaseViewBeanInfo.java	4

LocationTypes.java	4
SecurityPolicy.java	4
TrialCaseDetailData.java	4
MaskElement.java	4
MaskException.java	4
PropertyStrategy.java	4
ValidationException.java	4
SQLHelper.java	4
CacheServlet.java	4
OCOHome.java	4
ProceedingHistoryHome.java	4
ScreenHome.java	4
CopHome.java	4
TrackHistory.java	4
OOTHome.java	4
TrialCaseInfoHome.java	4
ScjHome.java	4
TrialCaseFillingDateHome.java	4
SenHome.java	4
ConnectionBuilder.java	4
DeletionFailureException.java	4
IncompleteObjectException.java	4
InsertionFailureException.java	4
InvalidJISUserException.java	4
InvalidPasswordException.java	4
InvalidTokenException.java	4
NoDataFoundException.java	4
NoDefaultParticipantException.java	4
NoSessionFoundException.java	4
UpdateFailureException.java	4
ActionCode.java	4
ActionCodeHome.java	4
Code.java	4
CodeHome.java	4
EventOpinion.java	4
SessionManager.java	4
ServerProxyCMPHome.java	4
SuperiorDocketManager.java	4
ChangeTrackProperties.java	3
ExceptionHandler.java	3
ManageParticipantsViewBeanInfo.java	3
SPExecutor.java	3
SPExecutorHome.java	3

CalendarManagerHome.java	3
OralArgumentManagerHome.java	3
ScreeningManagerHome.java	3
CaseManagerHome.java	3
ConsolidateCaseManagerHome.java	3
DocumentTemplateLocatorManagerHome.java	3
LinkingCaseManagerHome.java	3
TrackManagerHome.java	3
TrialCaseManagerHome.java	3
OACException.java	3
UnqualifiedConnection.java	3
EventManagerHome.java	3
EventsInfoHome.java	3
History.java	3
PersonManagerHome.java	3
PHN.java	3
CTC.java	3
DocumentGeneratorHome.java	3
SearchManagerHome.java	3
SecurityManagerHome.java	3
SessionManagerHome.java	3
SuperiorBasicManagerHome.java	3
SuperiorDocketManagerHome.java	3
SuperiorParticipantManagerHome.java	3
ServerSideClass.java	2
BoundPropertyChangeSource.java	2
UpperCaseValidator.java	2
ProceedingHistory.java	2
ExceptionNoDefaultParticipant.java	2
InvalidJabsUserException.java	2
InvalidJabsUserException.java	2
InvalidJISUserException.java	2
InvalidPasswordException.java	2
InvalidTokenException.java	2
NoDataFoundException.java	2
NoSessionFoundException.java	2
Total (1063 classes)	**128747**

APPENDIX B: CAPS ARCHITECTURE EVALUATION

Application Functionality

Following shows a high level use case diagram for the CAPS architecture

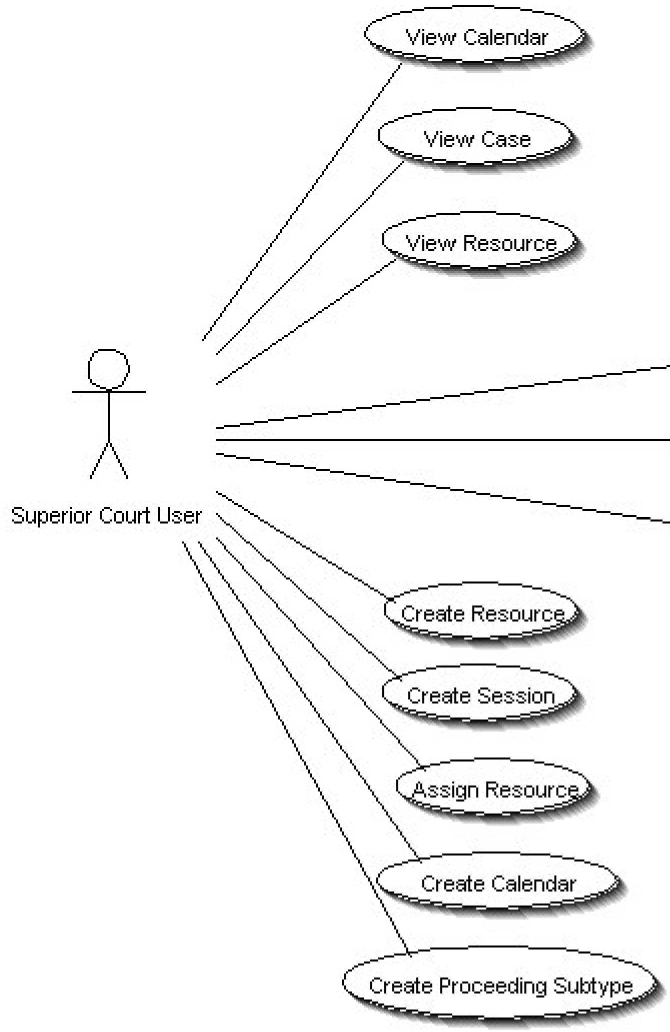


Fig. 1 CAPS User Case Diagram

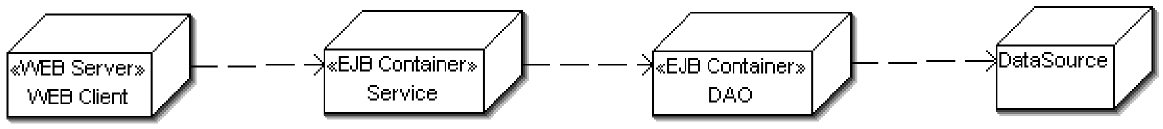


Fig. 2 CAPS High Level Deployment Diagram

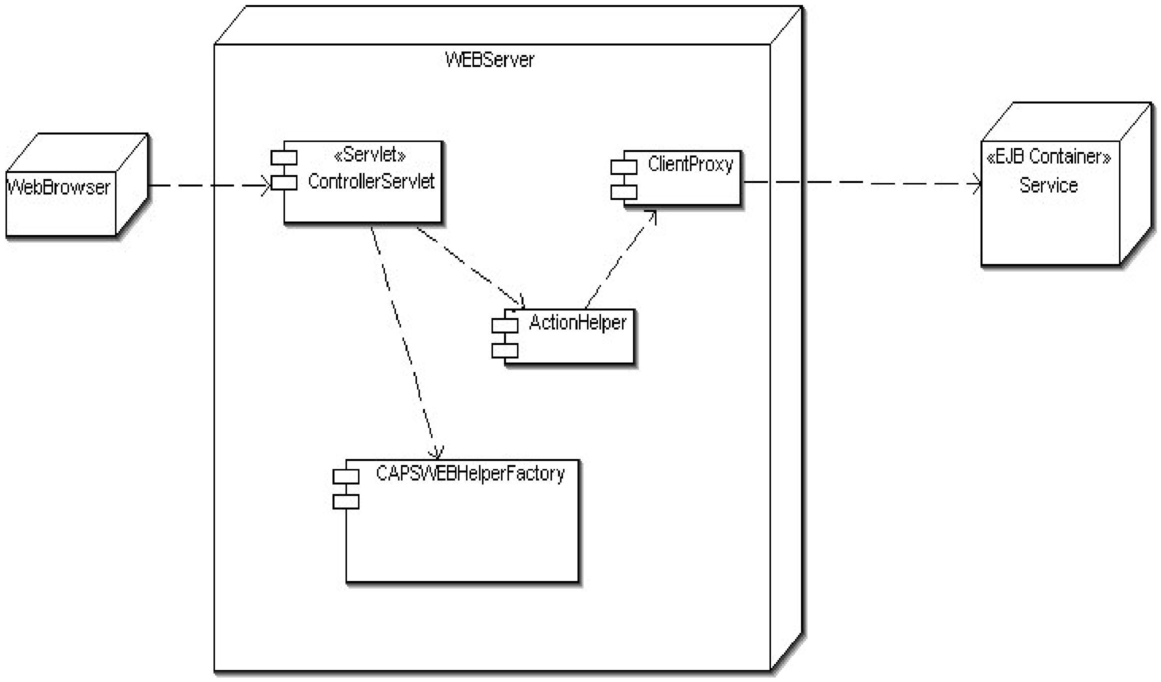


Fig. 3 CAPS WEB Server Deployment Diagram

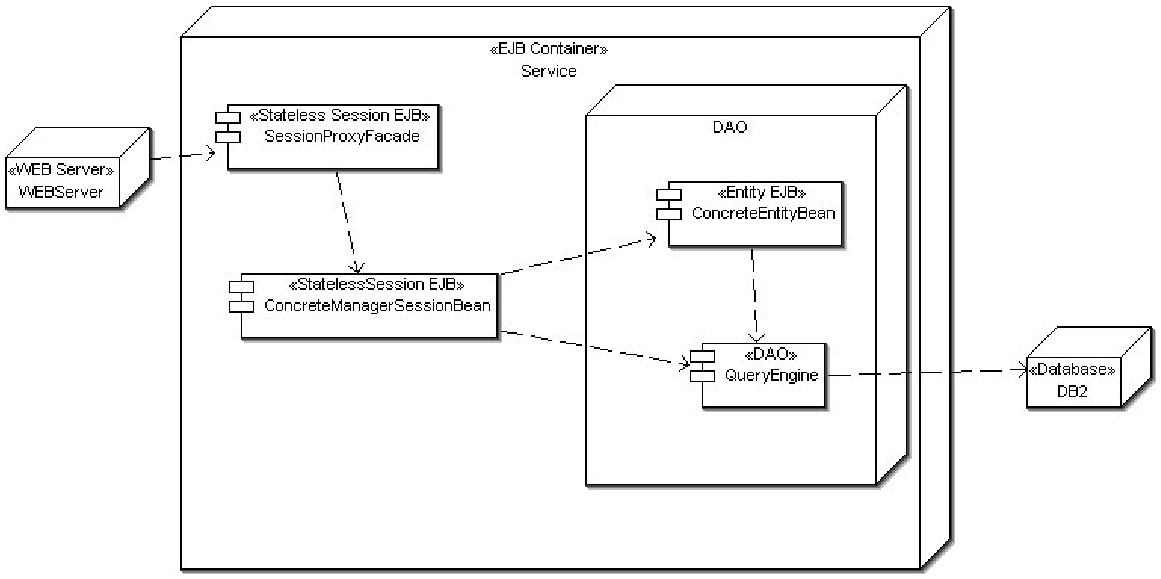


Fig. 4 CAPS Service Deployment Diagram

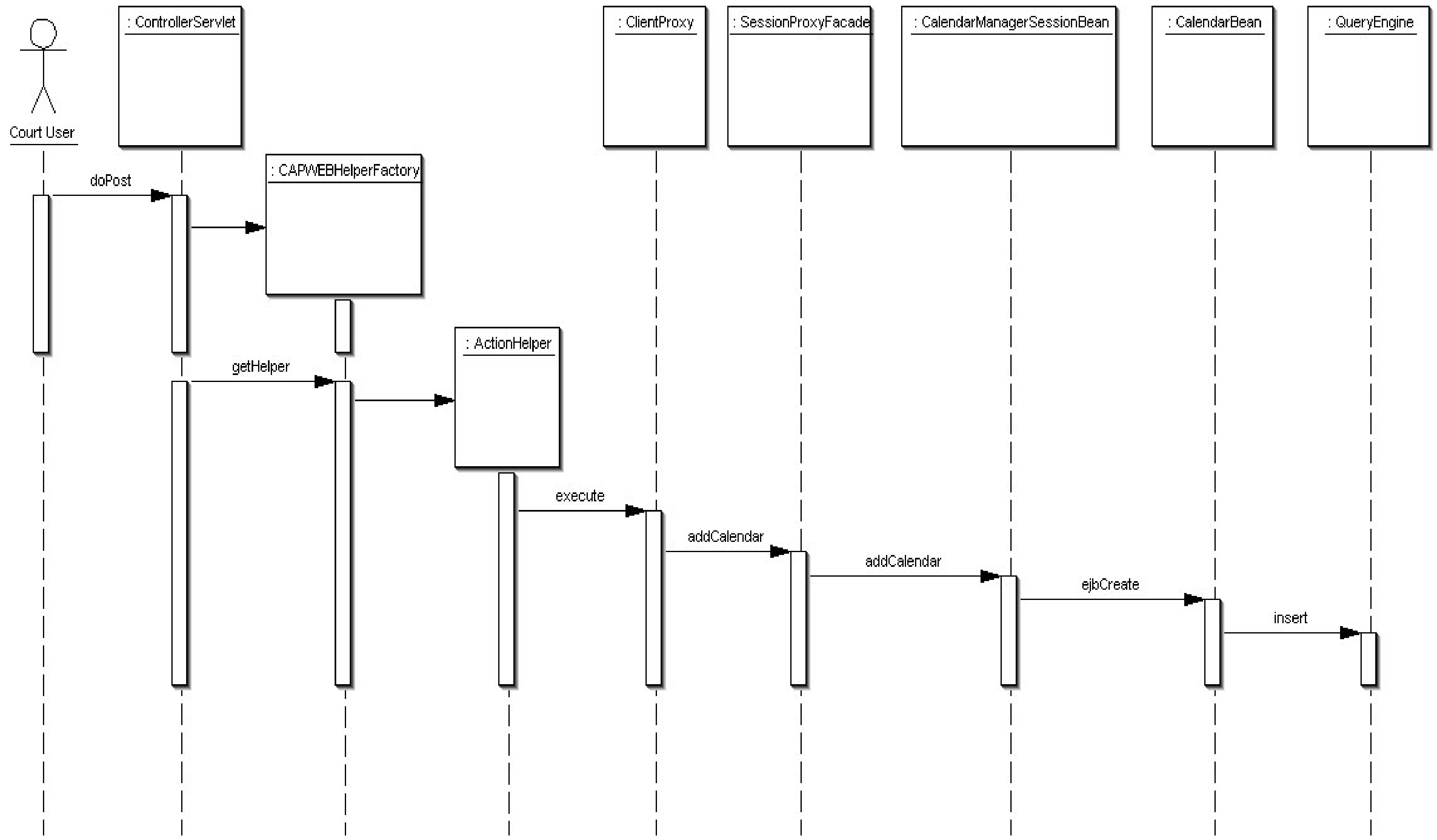


Fig. 5 CAPS Sequence Diagram For Add Calendar

Architecture Issues

Server-intensive processing and communication overhead

Due to limitations of the current architecture, several requests must be sent from the client browser to the server when a single web page is rendered. Simple actions, such as the user selecting an item in a drop-down box, cause a round trip communication process with the server. The browser must send a request, the server must process it, the browser must receive the response, and then the browser must render the page again. This requires additional server processing resources, additional time, and can result in an interface which is slow to respond to user input.

EJB 1.1

The current implementation of CAPS uses the EJB 1.1 version which requires the use of remote interfaces for communication with the Enterprise Java Beans. This can impose additional processing overhead for the marshalling and unmarshalling of method parameters and return values and provides no value since the EJBs are deployed in the same process as the web application.
More recent versions of the EJB specification provide a local interface mechanism which prevents this overhead.

Business logic in stored procedures

Stored procedures are used to implement business logic for recurring schedule items. This can cause maintenance problems. Business logic should be centralized in the application architecture as much as possible. They are difficult to debug and are difficult to use in a development environment, impacting the productivity of the application developers.

JavaScript

JavaScript is currently used to do client-side user input validation such as field length checks. The JavaScript code exists in JSPs. These JSPs contain business logic along with presentation logic. This introduces maintenance problems, impacting developer productivity. This architecture also is difficult to test and reduces the effectiveness of the unit testing coverage. JavaScript also is difficult to maintain and debug because of variations between browsers. Application behavior can vary depending upon browser types, versions, and the users' browser settings. Additionally, if a user disables JavaScript using their browser because of security concerns, this application can not be run.

Complexity of certain EJBs

Some of ManagerBean is overly complex. For example, the ProceedingManagerBean has 5300 lines of code. This negatively affects developer productivity.

Entity Bean overhead

Currently CAPS uses coarse-grained EJB EntityBeans. Entity Beans provide some benefit but impose a large amount of overhead. The benefits include declarative transaction and security management, as well as concurrency and caching services. However, the CAPS application does not take advantage of much of these services. The overhead imposed by using Entity Beans impacts both runtime performance and developer productivity. They are slow to develop. They require the implementation of several classes and deployment descriptors, they rely on a separate compilation process, and they require the developer to re-deploy the application. They are difficult to test because they require the services provided by the WebSphere server to run.

Scattered business rules
There is no central place to perform business rule validations. Those business rules are scattered all over the ManagerBeans, which contain much duplicate code. This causes maintenance problems.

Fine-grained project organization
There are 28 projects including 26 EJB projects, and 2 Web projects. Each EJB is contained in its own EJB project.

Source tree difficult to manage in development environment
WSAD project files have not been added to the perforce source tree. So it is very difficult to import the source java files in to WSAD workspace.

Merits of CAPS architecture

Simple and Layered Architecture
The CAPS architecture generally maintains a separation of concerns in various layers so it is easily maintainable.

JavaDocs standards followed
The code has adequate documentation.

Course-grained entity beans
Uses coarse-grained EntityBeans instead of fine-grained EntityBeans which greatly enhances maintainability.

Data Access Objects
The Entity Beans delegate their persistence logic to Data Access Objects, which are easier to maintain, can be tested more efficiently, and allow for a looser coupling between the application code and the database schema.

Ability to address multiple court levels
There are 2 session beans, viz. CaseInformationManager and ParticipantInformationManager, which address the issue of retrieving data from multiple court levels. These beans have generic methods, “getCaseInformation” and “getAllParticipants” respectively, which accept “COURTCONSTANTS” as input, and based on that return the correct data. Thus, these beans provide a way of factoring out vertical segments of functionality, managing cases, managing events and managing participants.

Source Lines of Code (excluding comments and white spaces)

CLASS NAME	TOTAL LINES
ProceedingQueryEngine.java	3611
CalendarSessionQueryEngine.java	2577
ProceedingManagerBean.java	2326
SessionSetupHelper.java	2197
CalendarManagementHelper.java	2082
SetProceedingHelper.java	2048
ProceedingSubtypeHelper.java	1331
OutcomeHelper.java	1227
ProceedingSubtypeQueryEngine.java	1206
ResourceSetupHelper.java	1132
CalendarSessionManagerBean.java	1065
RecurrenceQueryEngine.java	948
AssignmentQueryEngine.java	927
SessionProxyManagerBean.java	917
ResourceAssignmentHelper.java	796
ResourceQueryEngine.java	794
OANInfoQueryEngine.java	714
CapsHelper.java	654
ResourceManagerBean.java	648
ProceedingSubtypeManagerBean.java	624
DateConverter.java	589
ClientProxy.java	560
CaseInfoQueryEngine.java	559
LoginHelper.java	545
AssignmentManagerBean.java	536
SessionExceptionHelper.java	533
CalendarSetupHelper.java	489
OutcomeQueryEngine.java	481
BatchHelper.java	478
OutcomeManagerBean.java	469
OANInfoHelper.java	467
ProceedingDetailHelper.java	457
ReschedulePendingHelper.java	440
CTCQueryEngine.java	436
CalendarQueryEngine.java	392
ResourceUnavailabilityHelper.java	387
RecurrenceResolver.java	347
OrganizationQueryEngine.java	344
CaseNumberFormatter.java	334
ScomisMappingHelper.java	324
CapsSessionProxyManagerBean.java	324
CalendarManagerBean.java	292

ProcessLevelSecurityImpl.java	273
OfficialQueryEngine.java	272
ServerHelper.java	240
OfficialManagerBean.java	237
ClientUtil.java	220
ParticipantInfoQueryEngine.java	216
RecurrenceUtil.java	213
CapsErrorMessageConstants.java	213
OrganizationManagerBean.java	203
AuthenticationQueryEngine.java	201
CaseInformationManagerBean.java	199
ServerValidator.java	198
DataCacheHolder.java	196
ProceedingBean.java	188
RecurrenceManagerBean.java	185
ScheduledProceedingBO.java	184
PersonFinder.java	182
P.java	176
AuthenticationManagerBean.java	170
CalendarManagementHelperConstants.java	167
ProceedingData.java	166
QueryConstants.java	163
CommonOutcomeData.java	162
OANInfoTO.java	156
RecurrenceData.java	155
SecurityQueryEngine.java	153
ScomisProceedingData.java	152
BaseCaseInformationData.java	149
SuperFileReaderImpl.java	149
TokenGenerator.java	148
DataValidator.java	143
CodeReaderImpl.java	136
ParticipantInformationManagerBean.java	129
MessageParser.java	125
PersonTO.java	125
ProceedingSubtypeData.java	124
CapsWebHelperFactory.java	122
SetProceedingHelperConstants.java	117
ConfirmationData.java	117
OacDateFormatter.java	116
UnavailabilityData.java	113
ResourceCategoryPersistentRepository.java	112
OANInformationManagerBean.java	112
SessionSetupHelperConstants.java	111
ProceedingExceptionalData.java	111
ErrorHolidayHolder.java	110

HolidayHolder.java	110
WarningHolidayHolder.java	110
ProceedingDetailData.java	110
ProceedingSubtypeBean.java	110
JndiContextHelper.java	110
TimeDTO.java	107
SessionProxyManager.java	104
ResourceData.java	102
SessionData.java	102
BaseOANInformationData.java	102
SessionTimeslotBO.java	101
EnvironmentUtils.java	101
JNDIHelper.java	100
ParticipantProceedingData.java	99
CaseTypeHolder.java	98
BaseCourtApplicationData.java	98
TransformManager.java	98
LogEntry.java	98
OANNumberFormatValidator.java	96
ProceedingSQLHelper.java	95
Masseuse.java	95
BaseCaseInformationDataComparator.java	95
ResourceCategoryHolder.java	93
SessionProceedingData.java	93
ProceedingSubtypeHolder.java	92
CapsHelperConstants.java	92
ResourceAssignmentData.java	92
UserData.java	92
CapsStaticDataHelper.java	90
SharedConstants.java	90
CaseCompletionData.java	90
CapsSessionProxyManager.java	88
OfficialSubtypeHolder.java	87
OutcomeHelperConstants.java	87
ProceedingSequencingData.java	86
DataTypeConvertor.java	86
DataTypeConvertor.java	86
OutcomeHolder.java	83
ResourceAssignmentHelperConstants.java	83
SessionSummaryData.java	83
ProceedingSubtypeHelperConstants.java	81
CTCBean.java	81
SerializeDataObjectToXml.java	81
ProceedingTypeHolder.java	80
SecurityContext.java	80
JDBCLogManager.java	80

AssignmentReasonHolder.java	78
BeanHomeHolder.java	77
MailHelper.java	77
OrganizationSubtypeHolder.java	75
ContextualAclEntry.java	75
StaticDataHolder.java	75
SecurityManagerBean.java	74
DateCalculator.java	74
OfficialTypeHolder.java	73
ProceedingTypeCaseTypeHolder.java	73
ResourceSetupHelperConstants.java	72
DateBasedRefreshCache.java	72
MultiSessionOutcomeData.java	71
TimeRangeDetailData.java	71
Resources.java	71
SessionExceptionHelperConstants.java	70
ResourceBean.java	70
UnavailabilityBean.java	70
StaticDataHelper.java	68
ParticipantTypeHolder.java	67
ResourceUnavailabilityHelperConstants.java	67
JSPUtil.java	66
CalendarBean.java	66
CalendarSessionBean.java	66
BaseCourtData.java	65
CourtDescriptionHolder.java	64
ProceedingDetailHelperConstants.java	64
ResourcePersistentRepository.java	64
AddressData.java	64
SessionOutcomeData.java	63
PersonData.java	63
ServletParameterParser.java	62
SimpleSessionData.java	61
ProceedingDetailDataComparator.java	61
LoggerBean.java	61
PermissionCollection.java	60
BaseResourceData.java	59
JudicialOfficerPersistentRepository.java	59
ParticipantData.java	59
CalendarSessionSQLHelper.java	58
ProceedingSubtypeKey.java	58
OrganizationPersistentRepository.java	57
SFCPRCNData.java	56
AccessControlEntry.java	56
SPLogManager.java	55
CaseTypeKeyHolder.java	53

ProceedingCommonData.java	53
ResourceServerData.java	52
ProceedingDTO.java	52
SimpleCalendarData.java	51
ScomisProceedingDataComparator.java	50
CapsDatabaseDataStore.java	49
CTCKey.java	49
SequencingHelper.java	48
ScomisMappingHelperConstants.java	47
SessionExceptionSummaryData.java	46
ResourceAssignmentSQLUtil.java	46
ConnectionInfo.java	46
BeanManagedEntityBean.java	46
ServletManager.java	46
BaseContentHandler.java	46
SessionSetupHelperErrorMessageConstants.java	45
ProceedingSubtypeSummaryData.java	45
ResourceNames.java	45
RowProperties.java	45
DomReader.java	45
CaseTypeClosedProceedingData.java	44
SessionResourcePersistentRepository.java	44
ParticipantTypeDataComparator.java	44
WASPooledConnectionBuilder.java	44
SecurityPolicy.java	44
ProceedingResourcePersistentRepository.java	43
BaseCapsDatesDataComparator.java	43
CompressCharacters.java	43
OANInfoHelperConstants.java	42
ResourceAssignmentDataComparator.java	42
PropertyInspector.java	42
ReschedulePendingHelperConstants.java	41
ProceedingSubtypeServerData.java	41
ProceedingManager.java	41
BaseCourtApplicationDataComparator.java	41
ResourceDataComparator.java	41
DataFormatter.java	41
BatchHelperConstants.java	40
CommonCalendarSessionData.java	40
ResourceCategory.java	40
ValidationUtils.java	40
ResourceType.java	39
DataTypeConvertor.java	39
EmailData.java	39
Validate.java	39
ParserManager.java	39

CalendarSessionManager.java	38
SFCSTATData.java	38
KeyValueFlagData.java	37
EnvironmentConstants.java	37
DatabaseSessionBean.java	37
BaseCapsDatesData.java	36
ReschedulePendingData.java	36
CaseTypeDataComparator.java	36
NameFormatter.java	36
Trace.java	36
KeyValuePairDataComparator.java	36
ProceedingSubtypeHelperErrorMessageConstants.java	35
LoginHelperConstants.java	34
ProceedingHistoryBean.java	33
SocialSecurityNumberFormatter.java	33
SelectedCourtsData.java	33
BaseControllerServlet.java	33
LogManagerBean.java	33
ViewData.java	32
AbstractResource.java	32
Format.java	32
BatchData.java	31
UserServerData.java	31
DateRange.java	31
ResourceCategoryList.java	31
SecurityConstants.java	31
SessionProceedingDataComparator.java	31
ResourceSetupHelperErrorMessageConstants.java	30
Court.java	30
ResourceDto.java	30
ConnectionProperties.java	30
OANInfoSQLHelper.java	30
ProceedingSubtypeManager.java	29
SubtypeData.java	28
CapsSecurityContext.java	28
BeanNameConstants.java	28
DateCompare.java	28
AccessControlContext.java	28
CalendarManagementHelperErrorMessageConstants.java	27
OutcomeHelperErrorMessageConstants.java	27
ResourceAssignmentHelperErrorMessageConstants.java	27
SetProceedingHelperErrorMessageConstants.java	27
KeyValuePairDataObject.java	27
ProceedingSubtypeSQLHelper.java	26
ServiceLocatorHelper.java	26
ByteArrayToken.java	25

AssignmentManager.java	25
CapsStaticDataHolder.java	24
CalendarSetupHelperConstants.java	24
ProceedingResource.java	24
UserPrincipal.java	24
ServiceLocatorHolder.java	24
SessionExceptionHelperErrorMessageConstants.java	23
ResAsgGroupRelationData.java	23
CacheSQLHelper.java	23
RecurrenceSQLHelper.java	23
AcordsException.java	23
QueryEngineAdapter.java	23
AclPermission.java	23
CapsWebHelperFactoryConstants.java	22
ResourceCategoryDto.java	22
ResourceTypeDto.java	22
AssignmentSQLHelper.java	22
ResourceManager.java	22
CapsServiceLocator.java	22
CaseInformationManager.java	22
PropertiesReader.java	22
PropertiesReader.java	22
OutcomeManager.java	21
DataAccessEngine.java	21
ApplicationException.java	21
SystemException.java	21
SingletonCacheHelper.java	21
CapsHelperErrorMessageConstants.java	20
ResourceCategorySearchCriteria.java	20
CalendarManager.java	20
SecurityPrincipal.java	20
ErrorMessageConstants.java	20
OANInformationManager.java	20
CapsCacheConstants.java	19
WebControllerServlet.java	19
ResourceGroupData.java	19
SessionResource.java	19
CourtDto.java	19
ResourceAccessLevel.java	19
Debug.java	19
CalendarSetupHelperErrorMessageConstants.java	18
CapsSecurityPolicy.java	18
Proceeding.java	18
ProceedingKey.java	18
RecurrenceManager.java	18
ScomisMappingHelperErrorMessageConstants.java	17

ResourceSQLHelper.java	17
CalendarSessionKey.java	17
ProceedingHistoryKey.java	17
UnavailabilityKey.java	17
CaseInfoSQLHelper.java	17
CodeReaderContentsImpl.java	17
ServiceLocator.java	17
OANNumberData.java	17
ResourceUnavailabilityHelperErrorMessageConstants.java	16
ByteArrayTokenDto.java	16
ResourceCategoryListDto.java	16
CalendarKey.java	16
ResourceKey.java	16
OfficialManager.java	16
OrganizationManager.java	16
OacHighLowDates.java	16
LoginHelperErrorMessageConstants.java	15
ProceedingDetailHelperErrorMessageConstants.java	15
AuthenticateException.java	15
SessionAdapter.java	15
FrameworkBaseException.java	15
XPathManager.java	15
XmlContentHandler.java	15
CalendarSessionViewData.java	14
BatchCaseData.java	14
ResourcePath.java	14
ProceedingSubtype.java	14
RowInfo.java	14
QueryEngine.java	14
EntityAdapter.java	14
ReschedulePendingHelperErrorMessageConstants.java	13
AssignmentGroupData.java	13
ClientConfirmationData.java	13
ParentChildIndexData.java	13
SessionExceptionData.java	13
ResourceTypes.java	13
SecurityProperties.java	13
OutcomeSQLHelper.java	13
ErrorMessageConstants.java	13
CodeReaderImpl_JDBC.java	13
DateKeyComparator.java	13
QueryEngineFactoryImpl.java	13
AppServerJndiConstants.java	13
BatchHelperErrorMessageConstants.java	12
OANInfoHelperErrorMessageConstants.java	12
ResourceTypePath.java	12

AuthenticationManager.java	12
CourtConstants.java	12
ImmutableUserData.java	12
DataStoreFactoryImpl.java	12
ParticipantInfoSQLHelper.java	12
ResourceCategoryPath.java	11
CapsSecurityException.java	11
InvalidValueException.java	11
RecordAlreadyExistException.java	11
RecordNotFoundException.java	11
ProceedingHistoryHome.java	11
BusinessObjectType.java	11
ImmutableQueryEngine.java	11
BaseData.java	11
RecordAlreadyExistException.java	11
RecordNotFoundException.java	11
ParticipantInformationManager.java	11
CalendarSessionData.java	10
Resource.java	10
CourtPath.java	10
PurificationException.java	10
CalendarSQLHelper.java	10
OfficialSQLHelper.java	10
ProceedingSubtypeHome.java	10
CalendarNameComparator.java	10
NoDVIParticipantException.java	10
DataInconsistencyException.java	10
DataModifiedException.java	10
FrameworkImplementationException.java	10
Timeslot.java	9
BasicResource.java	9
JudicialOfficer.java	9
Organization.java	9
ByteArrayTokenPath.java	9
ResourceCategoryListPath.java	9
ProceedingResourceRepository.java	9
ResourceRepository.java	9
SessionResourceRepository.java	9
AuthenticationSQLHelper.java	9
ProceedingHome.java	9
SecurityManager.java	9
InvalidParameterException.java	9
ImmutableBaseCourtData.java	9
CacheConstants.java	9
LogManager.java	9
LogManager.java	9

OANInfoQueryEngineErrorConstants.java	9
CourtValidator.java	8
DateRangeValidator.java	8
ResourceTypeSearchCriteria.java	8
JudicialOfficerRepository.java	8
OrganizationRepository.java	8
ResourceTypeRepository.java	8
CalendarHome.java	8
CalendarSessionHome.java	8
CTCHome.java	8
ResourceHome.java	8
UnavailabilityHome.java	8
QueryConstants.java	8
IncompleteInformationException.java	8
InvalidCaseTypeException.java	8
InvalidDateFormatException.java	8
InvalidLogonException.java	8
InvalidPersonTypeException.java	8
ScomisCaseOnlyException.java	8
DataStore.java	8
Logger.java	8
QueryConstants.java	8
SequencingHelperConstants.java	7
Fickle.java	7
OperationException.java	7
SecuritySQLHelper.java	7
SuperFileReader.java	7
ExceptionSubTypes.java	7
ExceptionTypes.java	7
AclContext.java	7
Helper.java	7
PersonComparator.java	7
ObjectUtil.java	7
DetailSessionViewData.java	6
DirtyDataException.java	6
CaseOfflineException.java	6
OrganizationSQLHelper.java	6
CodeReader.java	6
PropertiesNotFoundException.java	6
InvalidBusinessStateException.java	6
BeanManagedEntity.java	6
EJBConstants.java	6
DataStoreAccessException.java	6
ParameterNotFoundException.java	6
AccessLevel.java	6
ErrorMessageConstants.java	6

RollbackException.java	5
ResourceSQLUtil.java	5
ResourceCategoryRepository.java	5
JudicialOfficerSQLUtil.java	4
OrganizationSQLUtil.java	4
ResourceCategorySQLUtil.java	4
ConnectionBuilder.java	4
DeletionFailureException.java	4
IncompleteObjectException.java	4
InsertionFailureException.java	4
InvalidJISUserException.java	4
InvalidPasswordException.java	4
InvalidTokenException.java	4
NoDataFoundException.java	4
NoDefaultParticipantException.java	4
NoSessionFoundException.java	4
UpdateFailureException.java	4
DataStoreFactory.java	4
QueryEngineFactory.java	4
DataObject.java	4
ContextualPolicy.java	4
Policy.java	4
ServletConstant.java	4
ContentHandlerConstants.java	4
DynamicMapper.java	4
XmlConstants.java	4
Identifiable.java	3
ImmutableCapsSecurityPolicy.java	3
AssignmentManagerHome.java	3
AuthenticationManagerHome.java	3
CalendarManagerHome.java	3
CalendarSessionManagerHome.java	3
CapsSessionProxyManagerHome.java	3
OfficialManagerHome.java	3
OrganizationManagerHome.java	3
OutcomeManagerHome.java	3
ProceedingManagerHome.java	3
ProceedingSubtypeManagerHome.java	3
SessionProxyManagerHome.java	3
RecurrenceManagerHome.java	3
ResourceManagerHome.java	3
SecurityManagerHome.java	3
CaseInformationManagerHome.java	3
OACException.java	3
DataTransferObjectType.java	3
ImmutableSecurityPolicy.java	3

ServiceLocatorCacheConstants.java	3
HelperFactory.java	3
LoggerHome.java	3
LogManagerHome.java	3
OANInformationManagerHome.java	3
ParticipantInformationManagerHome.java	3
SequencingHelperErrorMessageConstants.java	2
Calendar.java	2
CalendarSession.java	2
CTC.java	2
ProceedingHistory.java	2
Resource.java	2
Unavailability.java	2
ExceptionNoDefaultParticipant.java	2
InvalidJabsUserException.java	2
InvalidJabsUserException.java	2
InvalidJISUserException.java	2
InvalidPasswordException.java	2
InvalidTokenException.java	2
NoDataFoundException.java	2
NoSessionFoundException.java	2
LogQueryEngine.java	2
ErrorMessageConstants.java	2
QueryConstants.java	2
TOTAL (526 classes)	**58710**

APPENDIX C: JIS ARCHITECTURE EVALUATION

The current JIS architecture was intended to provide a common framework which the JIS Migration project could be based on. This framework was developed by the AOC for a year or so and was expected to support the efforts of multiple development teams working concurrently in distributed locations. The goal of the framework was to ensure that these teams would be following a consistent development strategy as they implement new business functionality. This architecture was intended to evolve over a period of several years, becoming more refined as more use cases were implemented.

The architecture supports the development of a single desktop application that would be used by all court levels but would eventually provide personalized views for various types of individual users. It also provides a Service Layer that provides a common set of application services which can be consumed by the desktop application or by external applications using Web Services.

The architecture was designed to support agile development processes. Certain design decisions were made specifically to support unit testing. The entire framework was intended to support an iterative development process. The deployment and configuration models were intended to support continuous integration practices.

The New JIS architecture is described on the AOC intranet at http://inside.courts.wa.gov/jis/eaa/sad/index.cfm.

Current Project Status

The New JIS application is deployed in production, providing some Web Service integration with King County's systems. The desktop application currently provides only a very minimal set of functionality for filing simple docket entries. There is no current plan to deploy the desktop application into production. Additionally, there is no coherent database and data management strategy for the JIS Migration effort. There is also no security architecture built.

Architecture Details

Database Architecture

The legacy systems which the JIS Migration effort is intended to replace are all built on a single database. This database schema is extremely complicated. The JIS Migration plan anticipated that the data model would be evolved as business practices evolved to be more similar between various court levels. The New JIS architecture was designed to support a common data model shared by all courts.

A new database schema (NJISLOC) was designed and is used by the New JIS application development process, but no data migration or synchronization strategy was developed to support the deployment of this schema. The JIS application now uses some parts of the legacy database and some parts of the new database.

Layered Architecture

The JIS Architecture is a layered architecture, with different pieces of application functionality separated into different layers. This is intended to minimize the dependencies between various pieces of code and improve maintainability. The various layers serve to isolate the effects of change.

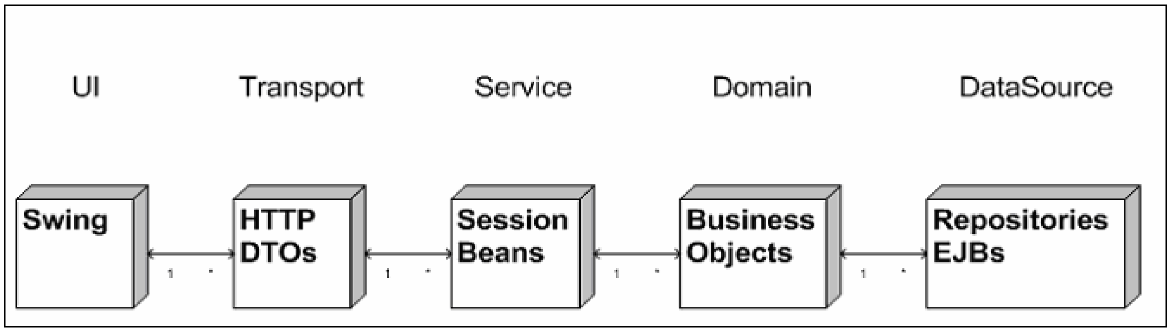


Figure 1 JIS Layers

Persistence Layer

The Persistence Layer provides the services which transfer data into and out of the databases and potentially other data stores. It is currently implemented with a variety of strategies. EJB 1.1 Entity Beans are used to insert and update data and JDBC/SQL is used for querying.

The Entity Beans require a simple one-to-one mapping of EJB to database table. Because of the complexity of the legacy database, this requires a very large number of entity beans to map to a single domain entity. The NJISLOC schema maps more closely to the domain model so the number of Entity Beans is fewer. However, the existence of two databases means that two separate sets of Entity Beans. Additionally, there is additional logic implemented which maps the two data models to the single domain model.

Domain Layer

The Domain Layer serves to provide a single model of the application's business logic. Domain layer services expose domain logic to service layer.

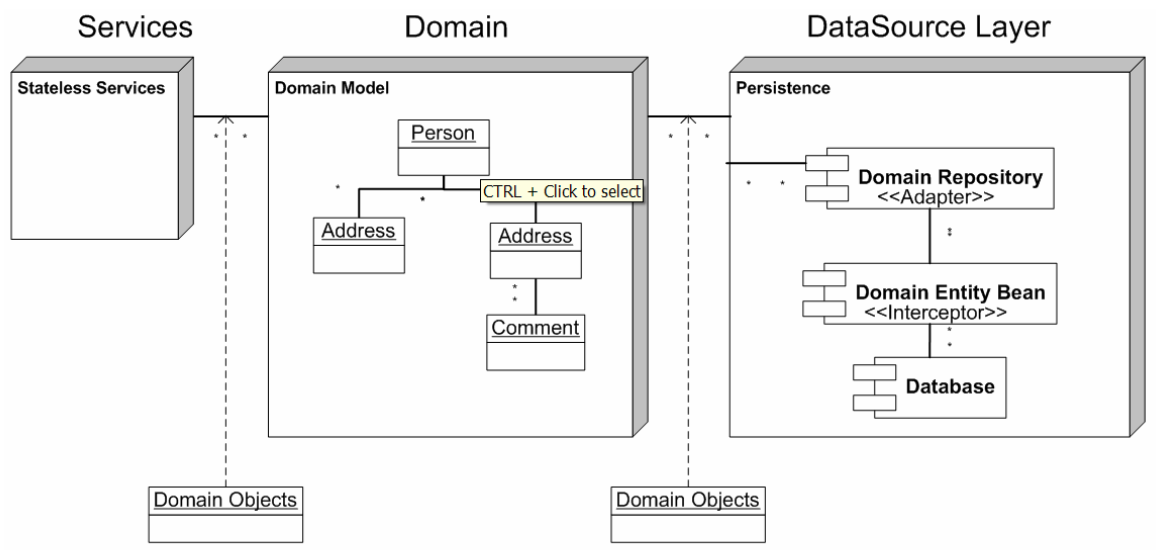


Figure 2 JIS Domain Layer

Service Layer

Services are exposed as Session Bean methods. The session bean infrastructure provides several features, but the only one that the JIS architecture uses is transaction management. The EJB container coordinates the transactions between this layer and the Entity Beans in the Persistence Layer. Session Bean method parameters are Data Transfer Objects.

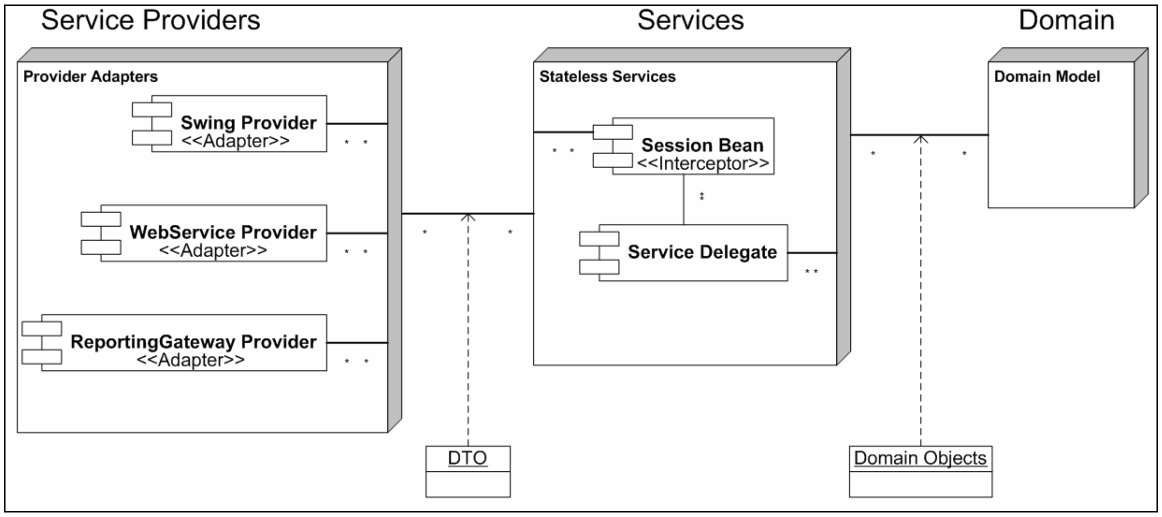

Service Providers
Services
Domain
Provider Adapters
Swing Provider <<Adapter>>
WebService Provider <<Adapter>>
ReportingGateway Provider <<Adapter>>
Stateless Services
Session Bean <<Interceptor>>
Service Delegate
Domain Model
DTO
Domain Objects

Figure 3 JIS Service Layer

Transport Layer

The Transport Layer is responsible for the communication between the client application and the server. A large portion of the JIS framework was written to support this layer. It requires application developers to implement request and response components for each service to be consumed by the client application. The developer also must implement Data Transfer Objects, classes which contain data that gets sent into and out of the service layer. The framework provides a mechanism for marshalling this data over HTTP and binding it to Presentation Layer components.

Web Services also are implemented using the Data Transfer Object architecture.

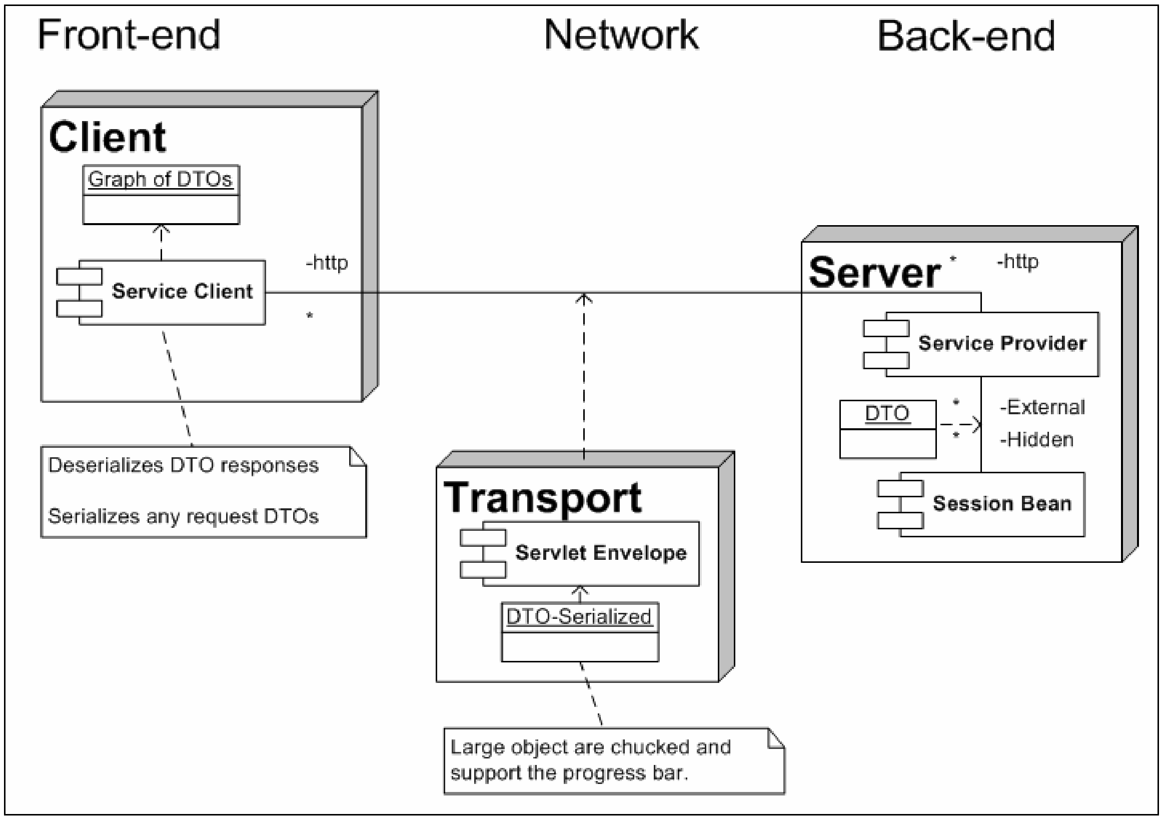


Figure 4 JIS Transport Layer

Presentation Layer

The Presentation Layer consists of custom Swing user interface components and GraphPath architecture. The GraphPath is intended to bind a portion of the data contained in the Data Transfer Objects to the custom Swing components.

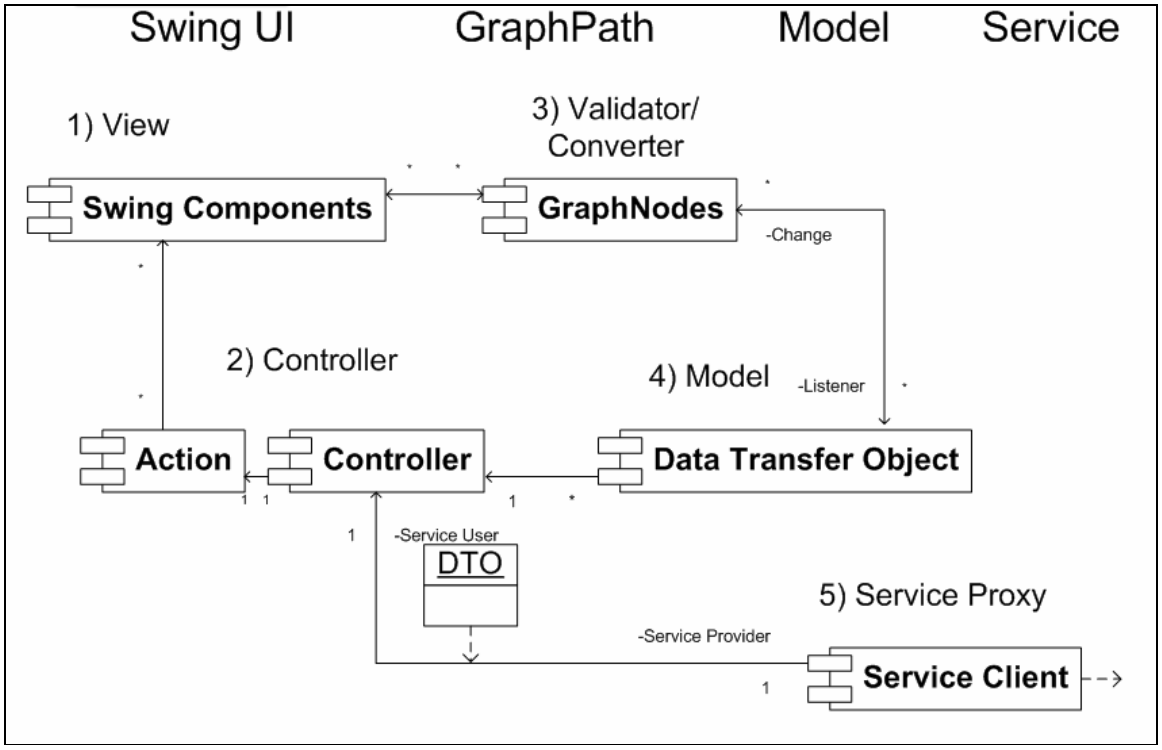


Figure 5 JIS UI Layer

Testing Architecture
The current JIS architecture and process rely on thorough test coverage for unit tests. The unit test suite currently consists of approximately 500 tests. The architecture provides a framework based on JUnit and JUnit extensions to allow for tests which are integrated with the continuous integration processes.

Security Architecture
The current JIS Architecture documentation addresses security issues but there is no support for authorization, authentication, or personalization in the current framework.

JIS Process
Team-based, iterative approach
The current JIS Application development process is a team-based iterative approach. The process provides successful mechanisms for communication between multiple distributed teams.

Continuous Integration
Continuous Integration is the practice of having a build and test process that provides constant verification of the state of the code base. The JIS architecture is one single code base that is not branched or versioned. It is shared by several distributed teams. If one team member checks in code that prevents the application from building, it can drastically affect the productivity of all teams. A continuous integration process minimizes this risk.

The continuous integration process for the current JIS application is a set of Ant scripts which are designed to run nightly on a server at AOC.

New JIS Limitations

1. Database
It is required that the New JIS application allow users to access data which is maintained by legacy applications. It is also required that data modified with the New JIS Application be viewed in the legacy applications. There has been no database architecture designed to address these difficult requirements.

1.1. The legacy data model is very complex and it does not match the New JIS application's domain model
The legacy schema supports each of the court layers with different entities but the New JIS Application treats each court layer the same. This dramatically slows the pace of development for the New JIS application. The application developer is forced to add persistence logic for three separate structures and then combine them into one new model. This effectively quadruples the amount of code which must be written when new persistence logic is needed. A database architecture could be designed which would provide a simplified interface which would isolate much of this complexity from the application developer.

1.2. The legacy schema relies on a mechanism for creating identifiers that can only be used on the mainframe
The JIS architecture does not provide a strategy for inserting new data in the legacy database. A new database architecture must not rely on any functionality that can only exist on the mainframe.

1.3. The new NJISLOC schema doesn't match the new JIS application requirements
It hasn't evolved through the same agile process as the code base. It consists mostly of structures that were copied from the legacy schema. Because these structures don't map closely to the JIS application's domain model, it complicates the implementation of new business logic. A new database architecture is necessary to isolate the application developer from this complexity.

1.4. The databases aren't maintained well with the continuous integration system and build system
The artifacts which maintain the database schema for the mainframe production system are not maintained with the build and continuous integration system. They can easily become out of synch, causing problems which aren't found until deployment time. A new database architecture should decouple the artifacts necessary for maintaining the production system from those that maintain development databases.

1.5. The database naming conventions make the database structures difficult to understand
Table and column names are not easily human readable. Application developers should be isolated from these legacy artifacts.

1.6. There is no strategy for managing test and sample data
Data used for unit testing is managed in an ad hoc fashion as new tests are created and maintained. There is no simple way to initialize the database with a testbed of data. There is no way to enter data in the legacy database for testing purposes without manually entering it through the various legacy applications. Additionally, no sample data has been provided for

acceptance testing and functional testing purposes. A new database architecture is necessary to simplify the management of testing data.

2. Persistence Layer

2.1. The variety of technologies increases the persistence layer complexity
The persistence layer uses both EJB 1.1 Entity Beans and JDBC with SQL statements. This requires developers to understand and maintain both of these mechanisms. More sophisticated Object/Relational mapping strategies exist which can remove much of this complexity.

2.2. Multiple datasources increase the persistence layer complexity
The persistence layer is responsible for synchronizing the data between the legacy database and the new schema as well as mapping the data model to the application domain model. This has not yet been implemented and will be very difficult to support. Each datasource requires its own EJB project and mapping, as well. A more efficient database architecture could provide a single view of the data from the application's perspective, allowing the application developer to focus on the JIS business domain logic.

2.3. One to one mapping with the database schema increases complexity
The Object/Relational mapping functionality provided by EJB 1.1 is primitive. It requires a single Entity Bean for each database table. A more sophisticated mapping technology could simplify this.

2.4. The Code/compile/test process is very slow
EJBs each require several files to be maintained. Whenever one of these is changed the entire project must be rebuilt and redeployed before unit tests can be run. Other alternatives exist that do not require this extra overhead.

2.5. EJBs complicate the build process
The Entity Beans require WebSphere to generate code before they can be deployed. They also have specific packaging requirements. Again, alternative Object/Relational mapping technologies do not have this overhead.

2.6. SQL is very complicated for accessing the legacy database
Because of the complexity and the difficult naming conventions, it is very difficult for application developers to write the necessary SQL to access the legacy schema. A more efficient database architecture could simplify this and an effective Object/Relational mapping strategy would generate the necessary SQL.

2.7. Tight coupling exists between the application code, deployment descriptors, and database schema
Small database changes can require a significant amount of application code to change. An Object/Relation mapping layer can provide a layer of abstraction which allows these types of changes to be isolated.

3. Domain Layer

3.1. The benefits of an isolated domain layer are not fully realized

The JIS architecture strives to encapsulate all business logic in a single layer of the application to reduce the implementation and maintenance costs. For this strategy to be effective, the framework must provide all of the services that the business logic requires. The framework provides only a rudimentary strategy for business rule validation. It provides no security or auditing infrastructure. A large amount of business logic must be implemented in the persistence layer due to limitations of the database and persistence architecture.

Because of limitations of the other layers, an application developer must provide a large amount of wiring code to make domain layer functionality useful. If the Transport and Persistence layers were simplified, the domain layer would be more useful.

3.2. The granularity of domain services is inconsistent

It is not clear whether CRUD (create, read, update, delete) operations should be exposed to clients as a single service or as multiple services. Various adapters have been developed to combine multiple service calls and map inconsistent domain concepts outside of the domain layer.

3.3. Implementing domain layer services requires complicated mappings between domain and transport mechanisms

The Data Transfer Object architecture is very complicated. Every service implementation requires a large amount of tedious code that an application developer must write to move data from Data Transfer Objects to Domain objects. The framework should handle this automatically.

4. Service Layer

4.1. The Session Bean code/compile/deploy/test process is very slow and complicates the build process

EJB Session Beans each require several files. Additionally, they require WebSphere to generate code. This complicates the build process unnecessarily.

4.2. The framework does not provide auditing, logging, or security for service requests

These are all important concerns which should be implemented in the service layer.

5. Transport Layer

5.1. The Data Transfer Object framework is very complicated

Application developers must write code to map data from Data Transfer Objects to Domain layer objects. This code is tedious, error-prone, and difficult to maintain. Additionally, the Data Transfer Object architecture is not intuitive and complicates the presentation layer code, as well. The framework should provide a Transport Layer that requires the application developer to do only minimal work to coordinate this communication between the client application and service layer.

6. Presentation Layer

6.1. The client platform JDK version does not match the development and the production environment

The targeted desktop platform for the JIS client application is JDK 1.4, but the development and deployment environments only support JDK 1.3. The development and deployment environments must be upgraded.

6.2. Transfer layer architecture makes client programming very slow

The UI programming is coupled with the transport mechanics, making UI development very slow. The framework must provide a simpler mechanism than the current Data Transfer Object architecture.

7. JIS application development process

7.1. The continuous integration process needs improvement

The continuous integration server is not maintained and there is no automated mechanism for notifying developers when the build has been broken. This is very important to prevent one team from impacting the productivity of another. There needs to be a simplified continuous integration that allows distributed teams to run their own continuous integration environment, as well.

7.2. It is difficult to build and deploy into different environments

The build process should make it easier to set up new environments.

8. Testing

8.1. Unit testing is difficult and coverage is not thorough

There are several limitations of the unit testing architecture. These are mostly due to the complexity of the various layers of the framework. EJBs are difficult to write unit tests for. Because there is no strategy for managing test data, each application developer must write code which manages setting up and restoring the environment before and after each test. Simpler mechanisms must be developed which allow test components in isolation.

8.2. No automated acceptance testing strategy has been defined

The JIS development effort was expected to have a corresponding testing effort which would produce automated acceptance tests which would verify each iteration goal. The architecture, however, does not define any strategy for writing and running these tests. The lack of an automated acceptance testing suite makes the application less stable. An acceptance testing strategy should cover testing of the service layer as well as the client application user interface.

8.3. No automated functional testing strategy has been defined

The JIS development also requires an automated suite to be used for functional and regression testing. There is no automated process to ensure that the JIS application behaves correctly from end to end.

8.4. No performance or scalability testing

No strategy has been defined to measure application performance.

8.5. No management of test data

Test data must be provided for automated unit tests, acceptance tests, and demonstration purposes.

Source Lines of Code (excluding comments and white spaces)

	TOTAL LINES
EJSLocalCMPCrt_a975ea9a.java	2001
EJSLocalCMPLaw_3609b123.java	1196
SubprojectSitePublisher.java	1171
EJSLocalCMPLwc_41c6fec8.java	986
CrtBeanFunctionSet_a975ea9a.java	971
MaintainMiscLawInformationPanel.java	700
LawBeanFunctionSet_3609b123.java	690
LawBeanFunctionSet_3609b123.java	656
TestPersistentCaseRepository.java	612
EJSLocalCMPDkt_1ab5b17a.java	601
LwcBeanFunctionSet_41c6fec8.java	556
ConcreteCrt_a975ea9a.java	518
PersistentCaseRepository.java	465
PersistentCaseDocketImpl.java	459
DktBeanFunctionSet_1ab5b17a.java	445
DktBeanFunctionSet_1ab5b17a.java	411
SearchForLawPanel.java	394
EJSLocalCMPLaw_0c803c3b.java	391
MaintainLegacyLawInformationPanel.java	386
TestManageLawServiceSessionBean.java	377
LawBeanFunctionSet_0c803c3b.java	366
TestManageCaseServiceSessionBean.java	360
TestInMemoryCaseRepository.java	342
ConcreteLaw_3609b123.java	336
CaseNumberFormatter.java	333
RequirementsModel.java	323
EJSLocalCMPCEA_c6bc28a7.java	321
LawInformationPanel.java	307
TestPersistentCredentialsRepository.java	303
CEABeanFunctionSet_c6bc28a7.java	300
MaintainLawInformationPanel.java	298
AddressBookPanel.java	298
CsBeanFunctionSet_bbd7c1f8.java	295
DocketDetailsPanel.java	294
ConcreteLwc_41c6fec8.java	294
FilterDecoratorTableModelTest.java	292
InMemoryCodeValueListRepository.java	290

TestElectronicFilingCredentialsServicesSessionBean.java	289
CrtBeanCacheEntryImpl_a975ea9a.java	287
EJSLocalCMPContact_9e696b51.java	286
EJSLocalCMPCs_bbd7c1f8.java	286
XMLSecurityServiceImpl.java	285
CmtBeanFunctionSet_0cb78212.java	285
CaseSearchController.java	280
TestCredentialValidationRules.java	279
PersistentLawRepository.java	274
DB2DDLConverter.java	273
CmtBeanFunctionSet_7e04ce8b.java	271
CEABeanFunctionSet_c6bc28a7.java	266
Law.java	264
CodeMetricsPublisher.java	264
CsBeanFunctionSet_bbd7c1f8.java	261
DocketPanel.java	258
TestDocketEntryWebService.java	258
JISInMemoryCodeValueListRepository.java	254
EJSLocalCMPCmt_7e04ce8b.java	251
CtcBeanFunctionSet_df1f44d1.java	251
SfcBeanFunctionSet_9aca3e38.java	251
EJSLocalCMPCmt_0cb78212.java	251
CmtBeanFunctionSet_0cb78212.java	251
ContactBeanFunctionSet_9e696b51.java	236
JCalendarComboBox.java	234
CrtBeanAdaptorBinding_a975ea9a.java	233
GraphPath.java	228
JCalendarPanel.java	227
LawBeanAdaptorBinding_3609b123.java	225
RequirementFactory.java	215
EfilingCredentialServiceSoapBindingStub.java	213
DataTransferObjectAdapter.java	212
BrowserController.java	211
PerBeanFunctionSet_1545092a.java	210
DocketEntryServiceBindingStub.java	208
ConcreteDkt_1ab5b17a.java	206
AbstractServiceProviderContainer.java	205
DocketEntryServiceDelegate.java	200
AddDocketEntryInformation.java	196
GUICaseSearchMainView.java	196
CrtBean.java	192
DktBeanAdaptorBinding_1ab5b17a.java	191
StubSearchForLawPanel.java	189
LawBeanAdaptorBinding_3609b123.java	189
Handler.java	183
TestLawDto.java	181
JXDDSchemaFormatter.java	180

LwcBeanAdaptorBinding_41c6fec8.java	177
LawBeanCacheEntryImpl_3609b123.java	177
LawBeanCacheEntryImpl_3609b123.java	177
ConcreteLaw_0c803c3b.java	176
PerBeanFunctionSet_1545092a.java	176
CEABeanAdaptorBinding_c6bc28a7.java	175
CsBeanAdaptorBinding_bbd7c1f8.java	173
AddressBookMainController.java	171
CmtBeanAdaptorBinding_0cb78212.java	171
SortDecoratorTableModelTest.java	169
GraphPathTest.java	166
TestObjectPath.java	165
DefaultIntrospectionAdapter.java	164
TestRequirementsModel.java	164
GUIAddressBookMainView.java	164
PerBeanAdaptorBinding_1545092a.java	163
ElectronicFilingCredentialServicesBeanBean.java	162
JUnitEEEntityBeanFunctionSet_3580e985.java	161
TestLawDomain.java	160
TestCRTEntityBean.java	156
TestDocketEntryDto.java	155
DktBeanAdaptorBinding_1ab5b17a.java	155
DocketEntry.java	154
TestLAWEntityBean.java	154
TestCaseNumberFormatter.java	153
SortDecoratorTableModel.java	152
TestXMLSecurityServiceImpl.java	152
JMonthPanel.java	151
LawBeanAdaptorBinding_0c803c3b.java	151
TestPersistentCaseDocketImpl.java	150
TestEfilingCredentialService.java	147
LwcBeanCacheEntryImpl_41c6fec8.java	147
MainPanel.java	145
TestDocketDetailsController.java	143
AbstractTransportTest.java	140
OverlayWindow.java	140
CmtBeanAdaptorBinding_7e04ce8b.java	139
CEABeanAdaptorBinding_c6bc28a7.java	139
CtcBeanAdaptorBinding_df1f44d1.java	137
SfcBeanAdaptorBinding_9aca3e38.java	137
CsBeanAdaptorBinding_bbd7c1f8.java	137
DecoratorTableModelFactoryTest.java	136
ContactBeanAdaptorBinding_9e696b51.java	135
ConcreteCmt_7e04ce8b.java	135
TestInMemoryLogEntryRepository.java	135
CmtBeanAdaptorBinding_0cb78212.java	135
GraphNodeButtonGroupModelTest.java	134

ConcreteCEA_c6bc28a7.java	134
ConcreteCs_bbd7c1f8.java	134
TestPersistentLawRepository.java	133
TestDocketEntryServiceDelegate.java	132
ExcelIterationPlan.java	132
CaseFactoryForTests.java	130
ObjectPath.java	129
StackTrace.java	127
PerBeanAdaptorBinding_1545092a.java	127
TestServerlessCaseSearchController.java	126
SortableTableColumn.java	126
CrtBeanInjectorImpl_a975ea9a.java	126
ConcreteCmt_0cb78212.java	126
JUnitEEEntityBeanAdaptorBinding_3580e985.java	125
TestSingleAttribute.java	124
TestAddressBookController.java	124
GUIMainView.java	123
TestCourtQueryObject.java	122
FilterDecoratorTableModel.java	122
TestManageCaseServiceSessionBeanPersistent.java	121
CaseSearchCriteria.java	120
ConcreteContact_9e696b51.java	120
EJSLocalStatelessManageContactService_30df9e30.java	120
EJSLocalStatelessElectronicFilingCredentialServic_a17b160f.java	120
AbstractDocbookTask.java	118
RequirementPackage.java	118
JISMainController.java	117
TestDataTransferObjectAdapter.java	117
TestBrowserController.java	116
CrtLocal.java	116
ManageCaseServiceBean.java	115
TestDocketEntryConverter.java	114
ExceptionContainer.java	114
EfilingAuthenticationServiceSoapBindingStub.java	113
ConcreteSfc_9aca3e38.java	113
GraphPathListSelectionModelTest.java	112
JUnitEETask.java	112
MainCasePanel.java	111
EJSLocalCMPJUnitEEEntity_3580e985.java	111
EJSLocalCMPPer_1545092a.java	111
DeleteEntryConfirmationDialogPanel.java	110
ListPropertyGraphNode.java	110
DateUtils.java	110
ListPropertyGraphNodeTest.java	109
DateValidatorAndConvertor.java	109
LwcBean.java	108
DocketDetailsController.java	107

GraphPathEventTest.java	107
ConcreteCtc_df1f44d1.java	107
JXDDSchema.java	106
BasicFilerIdentification.java	105
PopupTableCellEditorComponent.java	104
Contact.java	104
DocumentRenderer.java	103
ViewUtils.java	103
AbstractQueryObject.java	102
MaintainLawCommentsPanel.java	102
MockCaseSearchMainView.java	102
TestCaseSearchCriteria.java	101
TestServiceClient.java	101
CredentialServiceDelegate.java	99
DTOXMLReaderHelper.java	99
ChangeList.java	99
TestAddressBookMainView.java	99
JAutoTextField.java	98
TestLWCEntityBean.java	97
ManageLawServiceBean.java	96
JISPersistentCodeValueListRepository.java	95
GraphNodeTableModelTest.java	95
GraphPathListSelectionModel.java	95
AddDocketEntryAdditionalDetails.java	94
MSProjectHtmlPublisher.java	94
ConcretePer_1545092a.java	94
TestDocketDetailsView.java	93
GraphNodeCheckBoxModelTest.java	93
EJSLocalStatelessManageCaseService_fbda508d.java	93
DocketEntry.java	92
TestDKTEntityBean.java	92
DktBeanCacheEntryImpl_1ab5b17a.java	92
DktBeanCacheEntryImpl_1ab5b17a.java	92
EJSLocalCMPCmtHome_7e04ce8b.java	91
EJSLocalCMPCrtHome_a975ea9a.java	91
EJSLocalCMPLawHome_0c803c3b.java	91
EJSLocalCMPLwcHome_41c6fec8.java	91
EJSLocalCMPSfcHome_9aca3e38.java	91
EJSLocalCMPCmtHome_0cb78212.java	91
EJSLocalCMPCsHome_bbd7c1f8.java	91
EJSLocalCMPDktHome_1ab5b17a.java	91
EJSLocalCMPPerHome_1545092a.java	91
PerforceInfo.java	89
MockServiceClient.java	88
TestCONEntityBean.java	88
LawBean.java	88
PersistentCredentialsRepository.java	87

XMLUtils.java	87
TestUIComponentModelFactory.java	86
GraphJoinComboBoxModelTest.java	86
DTOContenHandlerTestHelper.java	85
ChangeReport.java	85
PerforceMissingFilesUtil.java	85
DefaultIntrospectionAdapterTest.java	84
JUnitXMLFormatter.java	84
CourtOfLimitedJurisdictionToolsPanel.java	83
SimplePropertyGraphNodeTest.java	83
RequirementsPublisher.java	83
ManageContactServiceBean.java	83
TestCaseNumberValidation.java	83
ExcelReleasePlan.java	82
LawBeanInjectorImpl_3609b123.java	82
LawBeanInjectorImpl_3609b123.java	82
CredentialValidationRules.java	81
CodeValue.java	81
DocketEntryBase.java	81
User.java	81
DocbookHtmlTransformer.java	81
ConcreteJUnitEEEntity_3580e985.java	80
TestManageContactServiceSessionBean.java	80
CaseIdentifierConverter.java	79
TableOverlayController.java	79
CrtBeanExtractor_a975ea9a.java	79
DocketEntryConverter.java	78
PersistentCourtRepository.java	78
TestDTOFactory.java	78
EJSCMPSfcHomeBean_9aca3e38.java	78
XMLTransformerTest.java	77
EJSCMPLawHomeBean_0c803c3b.java	76
EJSLocalCMPCtc_df1f44d1.java	76
EJSLocalCMPSfc_9aca3e38.java	76
EJSLocalStatelessConfigurationServiceBean_09339c44.java	75
TestManageLogEntryServiceBean.java	75
DktBean.java	75
CaseServiceProvider.java	74
MainTabbedPane.java	74
EJSCMPCmtHomeBean_7e04ce8b.java	74
TestCodeValueDto.java	73
TestJisMainController.java	72
TestDateUtils.java	72
EJSCMPCrtHomeBean_a975ea9a.java	72
EJSCMPLwcHomeBean_41c6fec8.java	72
LawBeanCacheEntryImpl_0c803c3b.java	72
TestVerifyTransactionManagementSessionBean.java	72

EJSCMPCmtHomeBean_0cb78212.java	72
EJSCMPCsHomeBean_bbd7c1f8.java	72
EJSCMPDktHomeBean_1ab5b17a.java	72
EJSCMPPerHomeBean_1545092a.java	72
TestAllDataTransferObjectAdapters.java	71
EJSLocalCMPJUnitEEEntityHome_3580e985.java	70
EJSLocalCMPContactHome_9e696b51.java	70
EJSLocalCMPCtcHome_df1f44d1.java	70
LwcBeanInjectorImpl_41c6fec8.java	70
EJSLocalCMPCEAHome_c6bc28a7.java	70
EJSLocalCMPLawHome_3609b123.java	70
LawLocal.java	70
TestCourtConverter.java	69
SuperiorCaseIndicator.java	69
Credentials.java	69
SimplePropertyGraphNode.java	68
PDFTransformer.java	68
LogEntryMessageSubscriber.java	67
TestAbstractQueryObject.java	65
StringVector.java	65
JISSession.java	65
JCalendarTestFrame.java	65
LawBean.java	65
TestPersistentCaseRepositoryForNJIS.java	65
ReportingGatewayServlet.java	65
AddDocketEntryPanel.java	64
TestJISSession.java	64
TestLAWEntityBean.java	64
StreamServiceProviderWorker.java	63
AppellateSuperiorToolsPanel.java	62
MaintainLawPanel.java	62
DuplicateCredentialsException.java	62
TestLogEntryFactory.java	61
TestEFilingAuthenticationService.java	61
AbstractAttribute.java	61
UIComponentModelFactory.java	60
LawDto.java	60
JIS.java	60
ServiceClientContext.java	60
TestCMTEntityBean.java	60
TestFilterDocketListAcceptance.java	59
GraphNodeListModelTest.java	59
CrtBeanCacheEntry_a975ea9a.java	59
LawPath.java	58
ManageCaseBusinessDelegate.java	58
GUILawSearchMainView.java	58
AbstractRepositoryFactory.java	58

GraphPathSelectionEventTest.java	58
DateValidatorAndConvertorTest.java	58
LwcLocal.java	58
CaseConverter.java	57
TestCaseIdentifierDto.java	57
CaseIndicator.java	57
CljCaseIndicator.java	57
ExpandedFilerIdentification.java	57
DocbookPDFTransformer.java	57
LawBeanExtractor_3609b123.java	57
LawBeanExtractor_3609b123.java	57
GUIDocketDetailsView.java	56
AuthenticationResult.java	56
CredentialUpgradeResult.java	56
SampleDTOFactory.java	56
TestJUnitEEEntity.java	56
WasisJournalRecord.java	55
ServiceClient.java	55
KeyServiceImpl.java	55
CaseSearchActions.java	54
TestCSEntityBean.java	54
DocketEntryDto.java	53
GUILawMaintenanceMainView.java	53
UtilityNavigationPanel.java	53
PDFTransformerTest.java	53
AbstractFilterPanel.java	53
CgiParser.java	53
CharacterConverter.java	53
TestCEAEntityBean.java	53
GraphNodeTableModel.java	52
Table.java	52
JISAuthenticationManager.java	52
TestJISAuthenticationManager.java	52
EJSLocalStatelessManageLawService_b76a53a7.java	52
TestCMTEntityBean.java	52
ManageContactBusinessDelegate.java	52
TestServiceRequestsAndProviders.java	52
CEABeanCacheEntryImpl_c6bc28a7.java	52
CEABeanCacheEntryImpl_c6bc28a7.java	52
CourtList.java	51
LawSearchCriteria.java	51
TestOrganizationQueryObject.java	51
TestQueryRunner.java	51
TestCaseDto.java	51
DocketEntryServiceServiceLocator.java	51
AuthenticationServiceServiceLocator.java	51
CredentialServiceServiceLocator.java	51

CodeValueList.java	51
GraphJoin.java	51
ProgressIndicator.java	51
ServiceLocator.java	51
LwcBeanExtractor_41c6fec8.java	51
TestContactDomain.java	51
EJSCMPCtcHomeBean_df1f44d1.java	50
JISSwingUtilities.java	49
MaintainLawBasicPanel.java	49
IncompleteIdentificationException.java	49
InvalidPasswordException.java	49
InvalidPinException.java	49
GraphPathBoundaryTest.java	49
FilterCriteriaRegistry.java	49
StringUtils.java	49
JAutoComboBox.java	49
EJSLocalStatelessJarDependencyDemoHome_593551ae.java	49
EJSLocalStatelessMultipleConnectionDemoHome_251eb65b.java	49
EJSLocalStatelessVerifyTransactionManagementHome_771efcdc.java	49
EJSLocalStatelessJUnitEEDemoHome_c4fdb533.java	49
EJSLocalStatelessManageContactServiceHome_30df9e30.java	49
CmtBean.java	49
EJSLocalStatelessConfigurationServiceBeanHome_09339c44.java	49
EJSLocalStatelessElectronicFilingCredentialServicHome_a17b160f.java	49
EJSLocalStatelessInMemoryRepositoryInitializerHome_af30a015.java	49
EJSLocalStatelessManageCaseServiceHome_fbda508d.java	49
EJSLocalStatelessManageCodeValueServiceHome_21b30c36.java	49
EJSLocalStatelessManageCourtServiceHome_bddd2e52.java	49
EJSLocalStatelessManageLawServiceHome_b76a53a7.java	49
EJSLocalStatelessManageLogEntryServiceHome_49a7d926.java	49
JISApplication.java	48
ContentHandlerProxy.java	48
IterationPlanPublisherValidation.java	48
Constants.java	48
TestSFCEntityBean.java	48
CsBean.java	48
DktBeanInjectorImpl_1ab5b17a.java	48
DktBeanInjectorImpl_1ab5b17a.java	48
MainActions.java	47
TestDocketEntry.java	47
GraphJoinTest.java	47
DTOXMLReader.java	47
HTMLTransformerTest.java	47
TestHttpTransport.java	47
CmtBeanCacheEntryImpl_7e04ce8b.java	47
CsBeanCacheEntryImpl_bbd7c1f8.java	47
CsBeanCacheEntryImpl_bbd7c1f8.java	47

QueryRunner.java	46
CaseHeaderPanel.java	46
ServiceRequest.java	46
TestChangeList.java	46
ManageLogEntryServiceBean.java	46
TestCaseSearchController.java	45
GUIDialogDisplayerView.java	45
CodeValue.java	45
TestStackTrace.java	45
Sample3Controller.java	45
CmtBean.java	45
TestNewJisMainController.java	44
Organization.java	44
TestWjrQueryObject.java	44
GraphNodeButtonGroupModel.java	44
AbstractDecoratorTableModel.java	44
RequiredFieldLabel.java	44
EJSCMPJUnitEEEntityHomeBean_3580e985.java	44
EJSCMPContactHomeBean_9e696b51.java	44
LawBeanInjectorImpl_0c803c3b.java	44
TestPEREntityBean.java	44
TestServiceLocator.java	44
EJSCMPLawHomeBean_3609b123.java	44
LawSearchItem.java	43
MockMainView.java	43
TestExcelProjectReleasePlan.java	43
PopupTableCellEditor.java	43
CaseNumberValidationRules.java	42
MockDocketDetailsView.java	42
LogEntry.java	42
TestFilterCriteriaRegistry.java	42
JISUrlInfo.java	42
KeySelectableComboBox.java	42
WaitGlassPane.java	42
ContactBeanCacheEntryImpl_9e696b51.java	42
EJSLocalStatelessJUnitEEDemo_c4fdb533.java	42
SfcBean.java	42
EJSLocalStatelessManageCodeValueService_21b30c36.java	42
TestManageContactBusinessDelegate.java	42
CmtBeanCacheEntryImpl_0cb78212.java	42
CmtBeanCacheEntryImpl_0cb78212.java	42
LogEntryFactory.java	41
TestDtoFactories.java	41
ListPropertyChangeSupport.java	41
DayLabel.java	41
EJSLocalStatelessManageLogEntryService_49a7d926.java	41
TestPersistentCodeValueListRepository.java	41

EFilingCredentialServiceTestDataFactory.java	40
ExecutionHandler.java	40
TestRequirementFactory.java	40
Utils.java	40
TestEncryptedDocumentImpl.java	40
ManageCodeValueServiceBean.java	40
CrudAddressDto.java	40
EJSCMPCEAHomeBean_c6bc28a7.java	40
DktBeanExtractor_1ab5b17a.java	40
DktBeanExtractor_1ab5b17a.java	40
DocketCode.java	39
DateTextFieldDocument.java	39
ComplexType.java	39
TraversableTable.java	39
ConfigurationServiceBeanBean.java	39
LoginProvider.java	38
CaseSearchMainView.java	38
GUIViewFactory.java	38
TestRepositoryFactoryBuilder.java	38
TestExcelReleasePlan.java	38
LawBeanExtractor_0c803c3b.java	38
WebServiceProxy.java	38
CEABean.java	38
OrganizationQueryObject.java	37
SuperiorCase.java	37
GraphPathInitializationOrderTest.java	37
TestListAttribute.java	37
ContainerStreamHandler.java	37
TestExcelIterationPlan.java	37
ProgressBarComponent.java	37
VerifyTransactionManagementBean.java	37
LawBeanCacheEntry_3609b123.java	37
Court.java	36
ViewFactory.java	36
RepositoryFactoryBuilder.java	36
TestTextLengthValidator.java	36
DTOTransformer.java	36
ServiceEvent.java	36
EditStateDecoratorTableModel.java	36
JISProperties.java	36
TestDB2DDLConverter.java	36
TestStringUtils.java	36
TestUtils.java	36
DktLocal.java	36
TestCaseConverter.java	35
PersistentCaseRepositoryUtil.java	35
WjrQueryObject.java	35

CaseIdentifierDto.java	35
MockViewFactory.java	35
AuthenticationServiceDelegate.java	35
CaseNotFoundSOAPException.java	35
DocketEntryNotFoundSOAPException.java	35
FieldsValidationSOAPException.java	35
InvalidCaseIdentifierSOAPException.java	35
MultipleCasesFoundSOAPException.java	35
GraphJoinComboBoxModel.java	35
TestCrudAddressList.java	35
ContactBean.java	34
Sample2Controller.java	34
Sample2Controller.java	34
DocketCodeList.java	33
AppelleteCaseIndicator.java	33
CaseIndicatorBase.java	33
MessagePublisher.java	33
ReportingGatewayClient.java	33
LocalServiceProviderRequestor.java	33
PropertiesManager.java	33
TestRequirementPackage.java	33
VisionDocument.java	33
TestSignedDocumentImpl.java	33
MultipleConnectionDemoBean.java	33
CtcBean.java	33
WebServiceXMLDocument.java	33
PerBean.java	33
BasicPersonalIdentifyingInformation.java	32
AddDocketEntryOKCancel.java	32
CmtBeanExtractor_7e04ce8b.java	32
CmtBeanInjectorImpl_7e04ce8b.java	32
CtcBeanCacheEntryImpl_df1f44d1.java	32
SfcBeanCacheEntryImpl_9aca3e38.java	32
TestManageCaseBusinessDelegate.java	32
CEABeanExtractor_c6bc28a7.java	32
CEABeanInjectorImpl_c6bc28a7.java	32
CEABeanExtractor_c6bc28a7.java	32
CEABeanInjectorImpl_c6bc28a7.java	32
DialogDisplayerController.java	31
ManageCaseBusinessDelegate.java	31
CodeValueDto.java	31
TestDocketActionDateRule.java	31
EditorDocument.java	31
SortableTableColumnModel.java	31
TestPropertiesManager.java	31
RootRequirementPackage.java	31
FlatButton.java	31

CtcBeanExtractor_df1f44d1.java	31
SfcBeanExtractor_9aca3e38.java	31
LwcBeanCacheEntry_41c6fec8.java	31
CsBeanExtractor_bbd7c1f8.java	31
CsBeanExtractor_bbd7c1f8.java	31
AcceptanceTestCase.java	30
UserConverter.java	30
Case.java	30
LawSearchServiceProvider.java	30
GraphSelectionEvent.java	30
TestImmutableRule.java	30
MessageSubscriber.java	30
DecoratorTableModelFactory.java	30
RawComplexType.java	30
ContactBeanExtractor_9e696b51.java	30
CtcBeanInjectorImpl_df1f44d1.java	30
SfcBeanInjectorImpl_9aca3e38.java	30
TestWebService.java	30
Test2JUnitEEDemo.java	30
CmtBeanExtractor_0cb78212.java	30
CsBeanInjectorImpl_bbd7c1f8.java	30
CmtBeanExtractor_0cb78212.java	30
CsBeanInjectorImpl_bbd7c1f8.java	30
PersistentLogEntryRepository.java	29
ContainerException.java	29
EJSLocalStatelessVerifyTransactionManagement_771efcdc.java	29
EJSLocalStatelessManageCourtService_bddd2e52.java	29
Sample3View.java	29
TestManageCodeValueServiceSessionBean.java	29
DocketEntryPath.java	28
BusinessDelegateFactory.java	28
ContainerRuntimeException.java	28
LookupTable.java	28
ExcelProjectReleasePlan.java	28
RequirementTypes.java	28
ContactBeanInjectorImpl_9e696b51.java	28
ManageCourtServiceBean.java	28
TestManageCourtServiceSessionBean.java	28
HttpServiceProviderContainerServlet.java	28
CmtBeanInjectorImpl_0cb78212.java	28
CmtBeanInjectorImpl_0cb78212.java	28
DtoFactoryFactory.java	27
AuthorizationLevelValues.java	27
CodeValueListDto.java	27
IntrospectionAdapter.java	27
ReportType.java	27
HttpServiceProviderRequestor.java	27

EditorDocumentTest.java	27
GraphNodeListModel.java	27
AbstractWorkerRunner.java	27
TestJUnitEEDemo.java	27
JISCodeValueServiceProvider.java	26
TestTable.java	26
DataTypeConvertor.java	26
AbstractType.java	26
JNDILocalEJBConstants.java	26
X509CertificateInfoImpl.java	26
PerBeanExtractor_1545092a.java	26
PerBeanExtractor_1545092a.java	26
MockServiceClientContext.java	25
InMemoryLogEntryRepository.java	25
DTOXMLReaderTest.java	25
HTMLTransformer.java	25
LogEntryMessageHandler.java	25
LocalServiceProviderContainerServer.java	25
GraphNodeCheckBoxModel.java	25
HistoryEntry.java	25
JUnitEEEntityBeanExtractor_3580e985.java	25
EJSStatelessJarDependencyDemoHomeBean_593551ae.java	25
EJSStatelessMultipleConnectionDemoHomeBean_251eb65b.java	25
EJSStatelessVerifyTransactionManagementHomeBean_771efcdc.java	25
EJSStatelessJUnitEEDemoHomeBean_c4fdb533.java	25
EJSStatelessManageContactServiceHomeBean_30df9e30.java	25
EJSStatelessConfigurationServiceBeanHomeBean_09339c44.java	25
EJSStatelessElectronicFilingCredentialServicHomeBean_a17b160f.java	25
EJSStatelessInMemoryRepositoryInitializerHomeBean_af30a015.java	25
EJSStatelessManageCaseServiceHomeBean_fbda508d.java	25
EJSStatelessManageCodeValueServiceHomeBean_21b30c36.java	25
EJSStatelessManageCourtServiceHomeBean_bddd2e52.java	25
EJSStatelessManageLawServiceHomeBean_b76a53a7.java	25
EJSStatelessManageLogEntryServiceHomeBean_49a7d926.java	25
ContactServiceGetContactsProvider.java	25
TestFrame.java	25
TestPersistentCourtRepository.java	25
ShowDocketDetailsAcceptanceTest.java	24
AbstractControllerTestCase.java	24
CourtConverter.java	24
DocketEntryListConverter.java	24
CourtQueryObject.java	24
TestUserDto.java	24
MaintainLawDetailPanel.java	24
JISTanTheme.java	24
PersistentCodeValueListRepository.java	24
JISDataValidationException.java	24

AbstractServiceProvider.java	24
TextFilterPanel.java	24
TestJISAuthenticationManagerWithInvalidUrls.java	24
ChainedRuntimeException.java	24
JUnitEEEntityBean.java	24
LawLocal.java	24
Sample3Frame.java	24
Sample1Frame.java	24
Sample2Frame.java	24
Sample2Frame.java	24
Sample2View.java	24
ClientRequestType.java	23
LawMaintenanceController.java	23
CodeValueConverter.java	23
Credentials.java	23
Person.java	23
TestDialogDisplayerView.java	23
ListAttribute.java	23
PDFReportGenerator.java	23
ReportingGateway.java	23
EditingCompleteInputVerifier.java	23
URLDecoder.java	23
JISLogin.java	23
AutoComboBoxEditor.java	23
EJSLocalStatelessJarDependencyDemo_593551ae.java	23
EJSLocalStatelessMultipleConnectionDemo_251eb65b.java	23
EJSLocalStatelessInMemoryRepositoryInitializer_af30a015.java	23
AddressBookMainView.java	23
HelpAction.java	23
TestDomainFactory.java	23
ContactServiceAddProvider.java	23
ContactServiceDeleteProvider.java	23
JisSubscriberMain.java	23
DialogDisplayerActions.java	22
User.java	22
DocketDetailsView.java	22
TestReportingGateway.java	22
AbstractFilterCriterion.java	22
TestSuitePackage.java	22
TextField.java	22
JUnitEEDemoBean.java	22
CmtKey.java	22
PerBeanCacheEntryImpl_1545092a.java	22
PerBeanCacheEntryImpl_1545092a.java	22
TestLawSearchMainView.java	21
DataTransferObjectType.java	21
IntrospectionAdapterBuilder.java	21

TestSuitePackage.java	21
BorderlessTextField.java	21
TestAbstractDecoratorTableModel.java	21
TestJISProperties.java	21
XMLSecurityFactoryImpl.java	21
TestDialogDisplayerController.java	20
TestSuitePackage.java	20
LawMaintenanceServiceProvider.java	20
TestPresentOrFutureDateRule.java	20
TestPresentOrPastDateRule.java	20
HTMLReportGenerator.java	20
ServiceEventListenerType.java	20
DateField.java	20
TestSuitePackage.java	20
TestXMLUtils.java	20
CtcKey.java	20
SfcKey.java	20
AddressBookControllerFactory.java	20
ContactServiceEditProvider.java	20
TestSuiteWebPackage.java	20
CEALocal.java	20
PerBeanInjectorImpl_1545092a.java	20
PerBeanInjectorImpl_1545092a.java	20
DktBeanCacheEntry_1ab5b17a.java	20
TestControllerInit.java	19
TestBasicFilerIdentificationDto.java	19
MainView.java	19
AddressDto.java	19
TextArea.java	19
LwcKey.java	19
AbstractServletTestCase.java	18
BasicFilerIdentificationPath.java	18
CaseIdentifierPath.java	18
MainModelListener.java	18
RepositoryFactory.java	18
ImmutableRule.java	18
SingleAttribute.java	18
TestDocumentRenderer.java	18
TestTableOverlayController.java	18
JUnitEEEntityBeanInjectorImpl_3580e985.java	18
ContactLocal.java	18
LawKey.java	18
ClientRequestType.java	18
CrudAddressTest.java	18
TestLegacyTokenIdentier.java	18
CsLocal.java	18
BasicFilerIdentificationDto.java	17

Attribute.java	17
DocketActionDateRule.java	17
TestPersonDto.java	17
XMLTransformer.java	17
ExceptionContainerFactory.java	17
CompositeInputVerifier.java	17
TestCharacterConverter.java	17
JUnitEEEntityBeanCacheEntryImpl_3580e985.java	17
Sample1Controller.java	17
CrudAddressPath.java	17
TestSuiteWebPackage.java	17
TestSuiteWebPackage.java	17
TestConfigurationSessionBean.java	17
LawSearchActions.java	16
LawSearchController.java	16
Paths.java	16
LegacyTokenIdentifier.java	16
JISUIConfig.java	16
CodeValueListRepository.java	16
IllegalPathException.java	16
ListPropertyGraphNodeEvent.java	16
AddressPath.java	16
ReportGeneratorFactory.java	16
AbstractServiceProviderRequestor.java	16
EditStateDecoratorModelTest.java	16
BooleanFilterCriterionTest.java	16
IntegerFilterCriterionTest.java	16
StringFilterCriterionTest.java	16
PropertyElement.java	16
EnsureNoUnaddedWorkspaceFiles.java	16
DefinedRequirementType.java	16
EncryptedDocumentImpl.java	16
CmtLocal.java	16
LawBeanCacheEntry_0c803c3b.java	16
RepositoryGenerator.java	16
CmtLocal.java	16
JISServiceClientContext.java	15
CaseIdentifierListConverter.java	15
JISCodeValueListRepository.java	15
TestCodeValueList.java	15
TestHTMLReportGenerator.java	15
ServiceProviderContext.java	15
ServiceProviderContainerContext.java	15
SessionType.java	15
TestSuitePackage.java	15
ToDo.java	15
AbstractPopupEditorComponent.java	15

SignedDocumentImpl.java	15
XEPFORenderer.java	15
InMemoryRepositoryInitializerBean.java	15
TestJarDependencySessionBean.java	15
TestMultipleConnectionSessionBean.java	15
AbstractRepositoryTestCase.java	15
TestSuiteWebPackage.java	15
AbstractConverterTestCase.java	14
AbstractCase.java	14
CaseDto.java	14
CourtDto.java	14
TestSuitePackage.java	14
MainModel.java	14
TestSuitePackage.java	14
ListEventConstants.java	14
PresentOrPastDateRule.java	14
PersonDto.java	14
TestAddressListDto.java	14
SimpleType.java	14
JUnitEEEntityKey.java	14
ContactKey.java	14
JarDependencyDemoBean.java	14
CrtKey.java	14
Sample2Panel.java	14
CmtKey.java	14
CsKey.java	14
DktKey.java	14
LawKey.java	14
PerKey.java	14
Comment.java	13
LegacyCase.java	13
CaseRepository.java	13
AuthenticationResultInformationDto.java	13
CredentialsDto.java	13
CredentialUpgradeResultInformationDto.java	13
LawSearchMainView.java	13
IntegerIdentifier.java	13
AttributeValidationRuleTestCase.java	13
NodeValuePresentationConverter.java	13
ListPropertyStateEvent.java	13
PresentOrFutureDateRule.java	13
AddressListDto.java	13
CommentDto.java	13
TestSuitePackage.java	13
TestPDFReportGenerator.java	13
ServiceEventEnvelope.java	13
ServiceRequestEnvelope.java	13

BooleanFilterCriterion.java	13
ChainedException.java	13
StreamPair.java	13
XMLSecurityFactory.java	13
Sample2View.java	13
EditAction.java	13
NewAction.java	13
CrudAddressListDto.java	13
TestIntegerIdentifier.java	13
TestSuiteWebPackage.java	13
TestCodeValueConverter.java	12
TestUserConverter.java	12
AbstractEntity.java	12
AuthorizationLevelTypes.java	12
DialogDisplayerValues.java	12
ExpandedPersonalIdentifyingInformation.java	12
DocketEntryListDto.java	12
TestCredentialsDto.java	12
UserDto.java	12
UserPath.java	12
ManageCaseBusinessDelegateClient.java	12
CodeValuePath.java	12
PersonPath.java	12
TestRequiredRule.java	12
TestStringLengthNonZeroRule.java	12
TestSuitePackage.java	12
IntegerFilterCriterion.java	12
StringFilterCriterion.java	12
PropertyElementInstance.java	12
RawElement.java	12
LocaleStrings.java	12
CancelAction.java	12
DeleteAction.java	12
SaveAction.java	12
TestSuiteWebPackage.java	12
TestSuiteWebPackage.java	12
CEABeanCacheEntry_c6bc28a7.java	12
AddUpdateDocketAcceptanceTest.java	11
LawSearchAction.java	11
CaseDtoFactory.java	11
CaseListDto.java	11
CodeValueDtoFactory.java	11
CourtDtoFactory.java	11
CourtPath.java	11
CredentialsPath.java	11
DocketEntryDtoFactory.java	11
TestCaseListDto.java	11

TestLawSearchDto.java	11
UserDtoFactory.java	11
CaseDocketServiceRequest.java	11
CaseListServiceRequest.java	11
CaseSaveServiceRequest.java	11
CaseServiceRequest.java	11
LawSearchServiceRequest.java	11
TextLengthValidator.java	11
TestMaximumStringLengthRule.java	11
DTOInputSource.java	11
HTMLGenerationException.java	11
PDFGenerationException.java	11
XMLFOTranslationException.java	11
XMLBindingException.java	11
MockBusinessDelegate.java	11
ClientInitRequest.java	11
TestSuitePackage.java	11
ThreadSafeRunner.java	11
RangeSimpleType.java	11
AbstractRequirementsTestCase.java	11
XMLSecurityService.java	11
CmtBeanCacheEntry_7e04ce8b.java	11
Sample1View.java	11
ContactServiceAddRequest.java	11
ContactServiceDeleteRequest.java	11
ContactServiceEditRequest.java	11
TestSuiteWebPackage.java	11
AbstractContainerTestCase.java	11
CsBeanCacheEntry_bbd7c1f8.java	11
ServletTestSuitePackage.java	10
TestSuitePackage.java	10
TestSuitePackage.java	10
QueryObjectImpl.java	10
LawRepository.java	10
CasePath.java	10
ExpandedFilerIdentificationDto.java	10
ExpandedFilerIdentificationPath.java	10
TestExpandedFilerIdentificationDto.java	10
CaseDocketServiceEvent.java	10
CaseListServiceEvent.java	10
CaseSaveServiceEvent.java	10
CaseServiceEvent.java	10
LawMaintenanceServiceRequest.java	10
LoginServiceRequest.java	10
InMemoryRepositoryInitializationProvider.java	10
TestGUICaseSearchMainView.java	10
TestJISUIConfig.java	10

TestSuitePackage.java	10
DocketEntryServiceBindingImpl.java	10
ChildObjectGraphNode.java	10
MaximumStringLengthRule.java	10
CommentPath.java	10
BusinessDelegateException.java	10
TestReportClient.java	10
TestTableModelListener.java	10
TestSuitePackage.java	10
ContactBeanCacheEntry_9e696b51.java	10
Sample3Action.java	10
Sample1Action.java	10
Sample2Action.java	10
Sample2Action.java	10
TestSuiteWebPackage.java	10
TestSuiteWebPackage.java	10
CmtBeanCacheEntry_0cb78212.java	10
DocketDetailsActions.java	9
MutableBoolean.java	9
AppellateCase.java	9
CLJCivilCase.java	9
CLJNonCivilCase.java	9
JISCodeValueList.java	9
CourtRepository.java	9
TestSuitePackage.java	9
CaseIdentifierListDto.java	9
CaseIdentifierListPath.java	9
CaseListPath.java	9
CourtListDto.java	9
CourtListPath.java	9
DocketEntryListPath.java	9
LawSearchDto.java	9
LawSearchPath.java	9
TestCourtListDto.java	9
ServletTestSuitePackage.java	9
DialogDisplayer.java	9
CodeValueListPath.java	9
TestSuitePackage.java	9
GraphNode.java	9
AddressListPath.java	9
TestSuitePackage.java	9
ReportGenerationException.java	9
ServiceRequestException.java	9
TestSuitePackage.java	9
TestSuitePackage.java	9
TestSuitePackage.java	9
TestLocalTransport.java	9

TestSuitePackage.java	9
ClientInitProvider.java	9
TestSuitePackage.java	9
FilterCriterion.java	9
JISAuthenticationException.java	9
NativeType.java	9
TestSuitePackage.java	9
KeyService.java	9
TestSuitePackage.java	9
XMLSecurityException.java	9
ManageContactServiceLocal.java	9
ElectronicFilingCredentialServicesBeanLocal.java	9
Constants.java	9
CrudAddressListPath.java	9
TestSuitePackage.java	8
EfilingCredentialServiceSoapBindingImpl.java	8
ServletTestSuitePackage.java	8
LogEntryRepository.java	8
TestCommentDTO.java	8
BusinessDelegate.java	8
DataValidationErrorServiceEvent.java	8
ServiceProviderRequestType.java	8
JISAuthenticationCancelledException.java	8
JISAuthenticationFailedException.java	8
JISNoSessionException.java	8
JISSessionException.java	8
EnumerationSimpleType.java	8
XMLDecryptException.java	8
TestSuitePackage.java	8
JUnitEEEntityLocal.java	8
CtcBeanCacheEntry_df1f44d1.java	8
SfcBeanCacheEntry_9aca3e38.java	8
ManageCaseServiceLocal.java	8
TestSuiteWebPackage.java	8
TestSuiteWebPackage.java	8
TestSuiteWebPackage.java	8
PerLocal.java	8
ServletTestSuitePackage.java	7
CLJCase.java	7
LawEnforcementAgency.java	7
TestSuitePackage.java	7
CredentialsRepository.java	7
GetOneIntQueryResultHandler.java	7
TestSuitePackage.java	7
PropertyKey.java	7
JISCodeValueServiceEvent.java	7
LawMaintenanceServiceEvent.java	7

LawSearchServiceEvent.java	7
InMemoryRepositoryInitializationEvent.java	7
TestLawMaintenanceMainView.java	7
TestSuiteWebPackage.java	7
TestSuiteWebPackage.java	7
RepositoryFactoryNotFoundException.java	7
TestSuitePackage.java	7
NodeValueValidator.java	7
TestSuitePackage.java	7
StringLengthNonZeroRule.java	7
ImpossibleToSatisfyRule.java	7
TestSuitePackage.java	7
TestSuitePackage.java	7
TestSuitePackage.java	7
ServletTestSuitePackage.java	7
LocalServiceProviderContainer.java	7
ClientDestroyProvider.java	7
InputValidationErrorHandler.java	7
TestSuitePackage.java	7
SwingRunner.java	7
TestSuitePackage.java	7
TestSuitePackage.java	7
TestSuitePackage.java	7
TestSuitePackage.java	7
GlassPaneContainerFrame.java	7
CertificateInfo.java	7
ManageLogEntryServiceLocal.java	7
DemoMessageService.java	7
TestSuitePackage.java	7
WebServiceProxyException.java	7
ServletTestSuitePackage.java	7
TestSuitePackage.java	7
TestSuitePackage.java	7
ContactServiceEvent.java	7
TestSuitePackage.java	7
TestSuiteWebPackage.java	7
TestSuiteWebPackage.java	7
HttpServiceProviderContainer.java	7
TestSuiteWebPackage.java	7
TestSuiteWebPackage.java	7
TestSuitePackage.java	6
TestCourt.java	6
InMemoryRepositoryInitializationRequest.java	6
TestManageCaseBusinessDelegateClient.java	6
LawMaintenanceMainView.java	6
CaseNotFoundException.java	6
CaseSearchException.java	6

DocketEntryNotFoundException.java	6
MultipleCasesFoundException.java	6
RepositoryFailureException.java	6
DtoStateConstants.java	6
NodeValidationException.java	6
CoreJavaTypePath.java	6
RequiredRule.java	6
SimplePropertyObservable.java	6
JISRuntimeException.java	6
ReportGenerator.java	6
JISLoginInfo.java	6
RequirementType.java	6
FORenderer.java	6
SignedDocument.java	6
CtcLocal.java	6
SfcLocal.java	6
ConfigurationServiceBeanLocal.java	6
ManageLawServiceLocal.java	6
PerBeanCacheEntry_1545092a.java	6
JISDocketActionCodesServiceRequest.java	5
JISDocketCodesServiceRequest.java	5
CaseBusinessDelegateParameters.java	5
DocketEntryServiceService.java	5
AuthenticationServiceService.java	5
CredentialService.java	5
CredentialServiceService.java	5
WebServicesUrls.java	5
InMemoryRepositoryFactory.java	5
PersistentRepositoryFactory.java	5
AttributeValidationRule.java	5
SimplePropertyGraphNodeEvent.java	5
NoSuchAttributeException.java	5
CookieParser.java	5
Type.java	5
TypeCreationListener.java	5
GlassPaneContainerType.java	5
EncryptedDocument.java	5
JUnitEEEntityBeanCacheEntry_3580e985.java	5
CmtLocalHome.java	5
CrtLocalHome.java	5
LawLocalHome.java	5
LwcLocalHome.java	5
SfcLocalHome.java	5
ManageCodeValueServiceLocal.java	5
UpcaseNodeValuePresentationConverter.java	5
ContactServiceGetContactsRequest.java	5
CmtLocalHome.java	5

CsLocalHome.java	5
DktLocalHome.java	5
PerLocalHome.java	5
LawQueries.java	4
PreparedQueryParameters.java	4
QueryResultHandler.java	4
DtoFactory.java	4
DuplicateCredentialsException.java	4
IncompleteIdentificationException.java	4
InvalidLogonIdException.java	4
InvalidPasswordException.java	4
InvalidPinException.java	4
DocketEntryService.java	4
EfilingAuthenticationServiceSoapBindingImpl.java	4
AggregateRoot.java	4
GraphSelectionEventListener.java	4
ListPropertyObservable.java	4
BooleanPath.java	4
DatePath.java	4
FloatPath.java	4
IntegerIdentifierPath.java	4
IntegerPath.java	4
StringPath.java	4
NullOutputStream.java	4
TransformationConfigurationException.java	4
Validateable.java	4
EditorView.java	4
RowTranslator.java	4
Condition.java	4
JNDIDataSourceConstants.java	4
AbstractAction.java	4
JUnitEEEntityLocalHome.java	4
ContactLocalHome.java	4
JUnitEEDemo.java	4
CtcLocalHome.java	4
ManageCourtServiceLocal.java	4
GuiFactory.java	4
CEALocalHome.java	4
LawLocalHome.java	4
JISSessionConstants.java	3
TableName.java	3
AuthenticationService.java	3
Identifier.java	3
Repository.java	3
ListPropertyGraphNodeListener.java	3
SimplePropertyGraphNodeListener.java	3
ListPropertyStateListener.java	3

MessageJNDIConstants.java	3
HttpTransportConstants.java	3
Editor.java	3
FilterRegistryChangeListener.java	3
TableName.java	3
WorkerType.java	3
Constants.java	3
NameType.java	3
GlassPaneContainerApplet.java	3
GlassPaneContainerDialog.java	3
JarDependencyDemoLocal.java	3
JarDependencyDemoLocalHome.java	3
JUnitEEDemoHome.java	3
MultipleConnectionDemoLocal.java	3
MultipleConnectionDemoLocalHome.java	3
VerifyTransactionManagementLocal.java	3
VerifyTransactionManagementLocalHome.java	3
ManageContactServiceLocalHome.java	3
ConfigurationServiceBeanLocalHome.java	3
ElectronicFilingCredentialServicesBeanLocalHome.java	3
InMemoryRepositoryInitializerLocal.java	3
InMemoryRepositoryInitializerLocalHome.java	3
ManageCaseServiceLocalHome.java	3
ManageCodeValueServiceLocalHome.java	3
ManageCourtServiceLocalHome.java	3
ManageLawServiceLocalHome.java	3
ManageLogEntryServiceLocalHome.java	3
Main.java	2
UndefinedRequirementType.java	2
JUnitEEEntityBeanInjector_3580e985.java	2
JUnitEEEntityBeanInternalHome_3580e985.java	2
JUnitEEEntityBeanInternalLocalHome_3580e985.java	2
ContactBeanInjector_9e696b51.java	2
ContactBeanInternalHome_9e696b51.java	2
ContactBeanInternalLocalHome_9e696b51.java	2
CmtBeanInjector_7e04ce8b.java	2
CmtBeanInternalHome_7e04ce8b.java	2
CmtBeanInternalLocalHome_7e04ce8b.java	2
CrtBeanInjector_a975ea9a.java	2
CrtBeanInternalHome_a975ea9a.java	2
CrtBeanInternalLocalHome_a975ea9a.java	2
CtcBeanInjector_df1f44d1.java	2
CtcBeanInternalHome_df1f44d1.java	2
CtcBeanInternalLocalHome_df1f44d1.java	2
LawBeanInjector_0c803c3b.java	2
LawBeanInternalHome_0c803c3b.java	2
LawBeanInternalLocalHome_0c803c3b.java	2

LwcBeanInjector_41c6fec8.java	2
LwcBeanInternalHome_41c6fec8.java	2
LwcBeanInternalLocalHome_41c6fec8.java	2
SfcBeanInjector_9aca3e38.java	2
SfcBeanInternalHome_9aca3e38.java	2
SfcBeanInternalLocalHome_9aca3e38.java	2
CEABeanInjector_c6bc28a7.java	2
CEABeanInternalHome_c6bc28a7.java	2
CEABeanInternalLocalHome_c6bc28a7.java	2
CmtBeanInjector_0cb78212.java	2
CmtBeanInternalHome_0cb78212.java	2
CmtBeanInternalLocalHome_0cb78212.java	2
CsBeanInjector_bbd7c1f8.java	2
CsBeanInternalHome_bbd7c1f8.java	2
CsBeanInternalLocalHome_bbd7c1f8.java	2
DktBeanInjector_1ab5b17a.java	2
DktBeanInternalHome_1ab5b17a.java	2
DktBeanInternalLocalHome_1ab5b17a.java	2
LawBeanInjector_3609b123.java	2
LawBeanInternalHome_3609b123.java	2
LawBeanInternalLocalHome_3609b123.java	2
PerBeanInjector_1545092a.java	2
PerBeanInternalHome_1545092a.java	2
PerBeanInternalLocalHome_1545092a.java	2
DataTransferObjectTranslator.java	0
TOTAL	**73522**
FILE COUNT	**1237**

APPENDIX D: ARCHITECTURAL COMPONENT COMPARISON

JIS

Purpose	Tool	Current Version (Date)
Java Development	Websphere Studio Application Developer (WSAD)	5.1.0.1 (10/1/2003)
Builds	Jakarta Ant	1.5.4 (10/1/2003)
Source Control	Perforce	2003.1 48707 (12/16/2003)
Java Development (J2SE) Environment	Java 2 Standard Edition from Sun (J2SDK)	1.4.2 (12/16/2003)
Java Development (J2EE) Environment	Java 2 Enterprise Edition from Sun(J2EESDK)	1.?.2 (12/16/2003)
Unit Test	JUnit, JUnitEE	
Database	DB2 on OS/390	
Application Server	IBM WebSphere Application Server	5.1

ACORDS

Purpose	Tool	Current Version (Date)
Java Development	Websphere Studio Application Developer (WSAD)	5.1.0.1
Builds	Jakarta Ant	1.5.4
Source Control	Perforce	2003.1 48707
Java Development (J2SE) Environment	Java 2 Standard Edition from Sun (J2SDK)	1.3.1
Java Development (J2EE) Environment	Java 2 Enterprise Edition from Sun(J2EESDK)	
Unit Test	JUnit, ROBOT testsuite	
Database	IBM DB2 on OS/390	7.2
Application Server	IBM WebSphere Application Server	5.1

CAPS

Purpose	Tool	Current Version (Date)
Java Development	Websphere Studio Application Developer (WSAD)	
Builds	Jakarta Ant	1.5.4
Source Control	Perforce	2003.1 48707
Java Development (J2SE) Environment	Java 2 Standard Edition from Sun (J2SDK)	1.4.2
Java Development (J2EE) Environment	Java 2 Enterprise Edition from Sun(J2EESDK)	
Unit Test	JUnit,	
Database	IBM DB2 on OS/390	7.2
Application Server	IBM WebSphere Application Server	

Stored Procedures used

Application	Stored Procedure Name	Purpose
ACORDS	JXSPSMF	Log Entry
	SP0005SX	Token Generation
	JXSPTK	
CAPS	EE_RECURRENCE	Recurrence Calculation
	EE_UNAVAILABILITY	
	EE_NEW_RCU	
	EE_RCUINS	
	EE_RCUCALC	
	EE_RCUCALC2	
	EE_UNAVCALC	

Count of Artifacts

Application	Total Java Files	Entity Beans	Session Beans	JSP
ACORDS	1064 (128747 SLOC)	67	19	77 (9971 SLOC)
CAPS	526 (58710 SLOC)	8	18	120 (12897 SLOC)

SLOC = Source Lines of Code, i.e. actual code, excludes white space, comments, formatting etc.

Comparative Analysis

Description	ACORDS	CAPS
User Interface	Swing based applet. Limited JSP Pages (not using any framework)	JSP pages.
Service Layer	Extra RMI Layer, and then Session Façade	Servlet calls over HTTP to session façade
Client Architecture	Home grown framework for Swing	Home grown framework similar to STRUTS.
JDK 1.4 compatible	No	Yes
Separation of Model / View / Controller	No.	Yes.
Remote EJB's	Yes	Yes
Session Beans	Stateless (Total=19)	Stateless (Total=18)
Entity Beans	Fine grained Container Managed Persistence (CMP) (Total=67)	Coarse grained Bean Managed Persistence (BMP) (Total=8)
Data Access	• Straight SQL calls from Session beans to DB for reads. • CMPs used for insert/update/delete	QueryEngine classes which are one per session bean. They have the persistent logic.
DB Access	Uses non-standard "Connection" classes in addition to the "datasource" provided by the WebSphere container	Strictly uses "datasource" of the container.
J2EE compliance	• Data Objects do not have clone(), toString(), and hashCode() methods.	Adhere to standards.
Database	DB2 on OS390	DB2 on OS390
Stored Procedures	JXSPSMF SP0005SX JXSPTK Used for Token Generation and Log Entry.	EE_RECURRENCE EE_UNAVAILABILITY EE_NEW_RCU EE_RCUINS EE_RCUCALC EE_RCUCALC2 EE_UNAVCALC Used for Recurrence Calculation.
	Repeatable_read	Repeatable_read_committed

APPENDIX E: NEW FUNCTIONALITY EFFORT ASSESSMENT

Overview
The first part of the document provides a rollup of the relative time costs for various types of common development efforts for each system's architecture. JIS is used for the current migration project while JIS NG is used for a new and optimized architecture for the migration project. Target is an industry standard that should be considered the optimum numbers that JIS NG is trying to achieve. The second part provides the underlying details that were used to provide the rollup numbers.

It should be noted that the JIS architecture is still in development and its numbers suffer from an incomplete and unpolished implementation that would become much cleaner and speedier to develop with if it were to be finished. Another point to note is that CAPS, while providing a simple and easy to develop upon architecture does not have the visual interface capabilities of ACORDS or JIS mainly due to technical limitations in the web JSP view layer. If CAPS had to support the full UI feature set of ACORDS or JIS its number would be much higher than the Swing equivalent due to the cost of supporting an extended UI feature set.

Target numbers are based on the analyst's experience with similar and actual components of the proposed architecture applied on projects with comparable types of business requirements. The project used as a basis for the industry Target is an existing highly evolved architecture optimized to make common tasks like Swing window creation very fast, the JIS NG architecture is not likely to fully achieve these results during its first year of use, but it should be used as the intended goal.

The issue for AOC is that to achieve these goals 20-30% of project time and personnel would have to be assigned to creating and maintaining this architecture. In addition, one senior developer-architect acting as mentor and architect would be needed for every ten developers. Several options for JIS NG will be promoted that do not assume this level of technical staffing and commitment so the numbers for JIS NG will represent a more reasonable combination of 10-20% of project time allocated to architecture and one senior developer-architect for every twenty developers.

Summary Numbers

Number of days required for a moderately experienced Java developer to learn the systems to the point where they can perform at the level assumed by the rest of the analysis.

	ACORDS	**CAPS**	**JIS**
Learning Curve	12 weeks	3-4 weeks	6-8 weeks

New Feature: Estimated days to add a create functionality for each application

	ACORDS	**CAPS**	**JIS**	**JIS NG**
Swing	23 days	20 days *	21.5 days	11.5 days
Web	25 days *	18 days	19.5 days *	11 days
WebService	26 days *	24 days *	24.5 days	19.5 days

* The client for this type does not exist yet. But it was estimated only for comparison.

Breakdown:

	ACORDS (Swing)	**CAPS (WEB)**	**JIS (Swing)**	**JIS NG**
Persistence Layer	4	6.5	4.5	1
Domain Layer	0	0	2	2
Data Transfer Layer	1	1	4	0
Service Layer	8	3.5	2.5	2.5
Transport Layer	1	0	1.5	0
Client Layer	9	7	7	6
Total	23	18	21.5	11.5

Maintenance: Estimated days to add a new field in the existing domain model

ACORDS	**CAPS**	**JIS**	**JIS NG**
9 days	3.75 days	6 days	2.5 days

Breakdown:

	ACORDS (Swing)	**CAPS (WEB)**	**JIS (Swing)**	**JIS NG**
Persistence Layer	1	1	2.5	0.5
Domain Layer	0	0	0.75	0.5
Data Transfer Layer	0.25	0.25	0.50	0
Service Layer	6	1	0.50	0.5
Transport Layer	0	0	0	0
Client Layer	1.75	1.5	1.75	1
Total	9	3.75	6.00	2.5

Enhancement: Estimated days to make a major change for an existing module
Requirement for Enhancement
If user tries to schedule an anchor case for hearing, schedule the consolidated cases also on the same day. Show a pop up window showing the cases that are consolidated and the time slots available on that day. User selects the cases and the time slot from the pop up. Application schedules all these cases for hearing on the same day on the specified time slots and shows them on the screen.

ACORDS	**CAPS**	**JIS**	**JIS NG**
18 days	11.5 days	18 days	9 days

Breakdown:

	ACORDS (Swing)	**CAPS (WEB)**	**JIS (Swing)**	**JIS NG**
Persistence Layer	0	4	4.5	0
Domain Layer	0	0	1.5	1
Data Transfer Layer	1	1	3.0	0
Service Layer	10	2.5	2.5	4
Transport Layer	1	0	1.0	0
Client Layer	6	4	5.5	4
Total	18	11.5	18.0	9

Detailed Analysis - Estimated days to add a brand new create functionality for each application

ACORDS
Add a new create CalenderSchedule functionality using SWING UI
CREATE (23 days)
Client side (10 days)
a) ValueObject (1 day)
   - Come up with a hierarchy of value objects
b) CalendarDetail (2 days)
   - Layout screen
c) CalenderView (6 days)
   - Applying validation rules and unit tests for any new validation rule (1 day)
   - Perform actions to call business delegate and unit tests (1 day)
   - Register UI components with ViewSupport and unit tests (1 day)
   - Pack/Unpack data on to screen components and unit tests (2 days)
   - Register View with Mainframe java class and unit test (1 day)
d) Business Delegate (1 day)
   - Update BusinessDegate(RMIClientProxy) interface, implementation and unit tests (1 day)

Server side (13 days)

a) RMI Layer (1 day)
   - Update RMIServlet interface, implementation and unit tests

b) Façade (2 days)
   - Update ServerProxyCMPBean session bean interface, implementation and unit tests. This class is very fragile and too big to handle with 9000 lines of code

c) CalendarManagerBean – SessionBean (6 days)
   - Implement business validation for createCalendar and unit tests. (3 days)
   - Update ServiceLocator and uiit tests (1 day)
   - Implement createCalendar functionality (2 days)

d) Calendar - Entity bean (4 days)
   - Create EntityBean (1 day)
   - Create EntityBean mapping to the database schema (1 days)
   - Unit test for entity bean (1 day)
   - Update ServiceLocator and uiit tests (1 day)

READ (23 days) Same amount of work required as CREATE
UPDATE (23 days) Same amount of work required as CREATE
DELETE (23 days) Same amount of work required as CREATE

Total estimated number of days for the CRUD calendar schedule = 92 days
Please note that this estimate only reflects the development time, but no analysis, QA tests, and deployment

Add a new getCalenderSchedule functionality using WEB UI
CREATE (25 days)

Client side (13 days)

a) ValueObject (1 day)
   - Come up with a hierarchy of value objects to be passed back and forth

b) JSP (12 days)
   - Layout screen (2 days)
   - Apply validation rules and unit tests (3 day)

     Note: Currently validations are done using javascriot which makes it very difficult for performing unit tests
   - Perform actions to call business delegate and unit tests (3 day)

     Note: There is no good framework been used here. All code is scattered inside the JSP, so difficult to develop
   - Pack/Unpack value objects on to screen components (2 days)
   - Update BusinessDegate(BCBean) interface, implementation and unit tests (1 day)
   - Update controller JSP (bridge.jsp) 1 day

Server side (12 days)

a) Façade (2 days)
    - Update ServerProxyCMPBean session bean interface, implementation and unit tests. This class is very fragile and too big to handle with 9000 lines of code

b) CalendarManagerBean – SessionBean (6 days)
    - Implement business validation for createCalendar and unit tests. (3 days)
    - Update ServiceLocator and uiit tests (1 day)
    - Implement createCalendar functionality (2 days)

c) Calendar - Entity bean (4 days)
    - Create EntityBean (1 day)
    - Create EntityBean mapping to the database schema (1 days)
    - Unit test for entity bean (1 day)
    - Update ServiceLocator and unit tests (1 day)

Currently JSP are been used by the public users and are designed to be READ ONLY. Folowing is the estimate for CREATE, UPDATE, REMOVE functionality if required
READ (25 days) Same amount of work required as READ
UPDATE (25 days) Same amount of work required as READ
DELETE (25 days) Same amount of work required as READ

Add a new createCalenderSchedule functionality using Webservice
CREATE (26 days)

WebService Layer (14 days)

a. Model and create XML schema's for Value objects (3 days)
b. Model and create XML schema's for Exception classes (1 day)
c. Create WSDL (3 days)
d. Generate java stub classes from WSDL (1 day)
e. Create CalendarWebserviceDelegate class. Implement createCalendarSchedule and unit tests (3 days)
f. Implement createSchedule in CaleandarWebserviceImpl(1 day)
g. Update global deployment descriptor and unit tests (2 days)

Server side (12 days)

a. Façade (2 days)
b. Update ServerProxyCMPBean session bean interface, implementation and unit tests. This class is very fragile and too big to handle with 9000 lines of code
c. CalendarManagerBean – SessionBean (6 days)
    - Implement business validation for createCalendar and unit tests. (3 days)
    - Update ServiceLocator and uiit tests (1 day)
    - Implement createCalendar functionality (2 days)
d. Calendar - Entity bean (4 days)
    - Create EntityBean (1 day)
    - Create EntityBean mapping to the database schema (1 days)
    - Unit test for entity bean (1 day)
    - Update ServiceLocator and uiit tests (1 day)

READ (26 days) Same amount of work required as CREATE

UPDATE (26 days) Same amount of work required as CREATE
DELETE (26 days) Same amount of work required as CREATE

CAPS
Add a new createCalenderSchedule functionality using WEB UI

Create 18 days

1) Client Layer (8 days)
    a) CalendarValueObject (1 day)
    b) CalendarHelper (2.5 days)
        - The controller class which has the implementation for the "CREATE" action. Processes the action and handles error conditions.
    c) CapsWebHelperFactory (0.5 day)
        - Update this to create/retrieve an instance of required Helper class
    d) Calendar.jsp (3 days)
    e) ClientProxy (1 day)
        - Update this to have the "create" method pass-through

2) Server Layer (6 days)
    a) SessionProxyManager (1 day)
        - Update the façade to delegate the "create" method call to the correct Session bean using Service Locator.
    b) CalendarManagerBean Total of 3 classes and 2 deployment descriptors (2.5 days)
        - Implement business validation and error messaging.
        - Call the service locator and execute the "create" method on the Calendar entity bean
        - Implement the roll back logic.
    c) Calendar Entity Bean – Total 3 classes and 2 deployment descriptors (2.5 days)
        - Implement the entity bean methods.
        - Provide the create method call to the QueryEngine.

3) Persistence/DAO Layer (4 days)
    a) CalendarQueryEngine (2 days)
        - Implement the method for "createCalendar", i.e. do the JDBC part to obtain connection, Create and Execute a Statement, close connection etc.
    b) CalendarSQLHelper (2 days)
        - The insert SQL for Calendar create.

READ (18 days) Same amount of work required as CREATE
UPDATE (18 days) Same amount of work required as CREATE
DELETE (18 days) Same amount of work required as CREATE

Add a new createCalenderSchedule functionality using WEB USING SWING UI
Create (20 days)

b) Client Layer (10 days)
    a) CalendarValueObject (1 day)
    b) CalendarDetail (2 days)
        - Layout screen
    c) CalenderView (6 days)
        - Applying validation rules and unit tests for any new validation rule been added (1 day)
        - Perform actions to call business delegate and unit tests (1 day)
        - Register UI components with ViewSupport and unit tests (1 day)
        - Pack/Unpack data on to screen components and unit tests (2 days)
        - Register View with Mainframe java class and unit test (1 day)
    c) ClientProxy (1 day)
        - Update this to have the "create" method pass-through.

c) Server Layer (6 days)
    a) SessionProxyManager (1 day)
        - Update the façade to delegate the "create" method call to the correct Session bean using Service Locator.
    b) CalendarManagerBean Total of 3 classes and 2 deployment descriptors (2.5 days)
        - Implement business validation and error messaging.
        - Call the service locator and execute the "create" method on the Calendar entity bean
        - Implement the roll back logic.
    c) Calendar Entity Bean – Total 3 classes and 2 deployment descriptors (2.5 days)
        - Implement the entity bean methods.
        - Provide the create method call to the QueryEngine.

d) Persistence/DAO Layer (4 days)
    a) CalendarQueryEngine (2 days)
        - Implement the method for "createCalendar", i.e. do the JDBC part to obtain connection, Create and Execute a Statement, close connection etc.
    b) CalendarSQLHelper (2 days)
        - The insert SQL for Calendar create.

READ (20 days) Same amount of work required as CREATE
UPDATE (20 days) Same amount of work required as CREATE
DELETE (20 days) Same amount of work required as CREATE

Add a new createCalenderSchedule functionality using WebService

Create (24 days)

1. WebService Layer (14 days)
    a. Model and create XML schema's for Value objects (3 days)
    b. Model and create XML schema's for Exception classes (1 day)
    c. Create WSDL (3 days)
    d. Generate java stub classes from WSDL (1 day)
    e. Create CalendarWebserviceDelegate class. Implement createCalendarSchedule and unit tests (3 days)
    f. Implement createSchedule in CaleandarWebserviceImpl(1 day)
    g. Update global deployment descriptor and unit tests (2 days)

2. Server Layer (6 days)
    a. SessionProxyManager (1 day)
        - Update the façade to delegate the "create" method call to the correct Session bean using Service Locator.
    b. CalendarManagerBean Total of 3 classes and 2 deployment descriptors (2.5 days)
        - Implement business validation and error messaging.
        - Call the service locator and execute the "create" method on the Calendar entity bean
        - Implement the roll back logic.
    c. Calendar Entity Bean – Total 3 classes and 2 deployment descriptors (2.5 days)
        - Implement the entity bean methods.
        - Provide the create method call to the QueryEngine.

3. Persistence/DAO Layer (4 days)
    a. CalendarQueryEngine (2 days)
       Implement the method for "createCalendar", i.e. do the JDBC part to obtain connection, Create and Execute a Statement, close connection etc.
    b. CalendarSQLHelper (2 days)
       The insert SQL for Calendar create.

READ (24 days) Same amount of work required as CREATE
UPDATE (24 days) Same amount of work required as CREATE
DELETE (24 days) Same amount of work required as CREATE

JIS
Add a new createCalenderSchedule functionality using SWING UI
CREATE (21.5 days)

Client side (11 days)

a. Create new ValidationRule if required (1 day)
b. Create CalendarDto, attach ValidationRules and unit tests (1 day)
c. Create ObjectPaths for CalendarDto and unit tests (1 day)
d. Create CalendarDtoFactory and unit tests (1 day)
e. Create View interface (1 day)
f. Create GUIView ,TestView and unit tests (3 days)
g. Create Controller and unit tests (3 days)

Server Side(10.5 days)

a. Create AddCalendarServiceRequest, CalendarServiceEvent and CalendarServiceProvider and unit tests – Servlet tier. (1 day)
b. Create Calendar domain object (1 day)
c. Create CalendarConverter that converts domain to dto and back and forth (1 day)
d. Create CalendarBusinessDelegate and implement createCalendarSchedule and unit test (0.5 days)
e. Create InMemoryCalendarRepository, implement createCalenderSchedule and unit tests (1 day)
f. Create PersistentCalendarRepository, implement createCalenderSchedule and unit tests (1.5 days)
g. Create CalendarManagerBean. Implement createCalenderSchedule with business validation and unit tests (2 days)
h. Create Calendar entity bean and map entity bean to the database schema (1 day)
i. Unit test for entity bean(1 day)
j. Add newly added beans to ServiceLocator (0.5 day)

READ (21.5 days) Same amount of work required as CREATE
UPDATE (21.5 days) Same amount of work required as CREATE
DELETE (21.5 days) Same amount of work required as CREATE

Add a new createCalenderSchedule functionality using WEB UI
CREATE (19.5 days)
Client side (9 days)

a. Create new ValidationRule if required (1 day)
b. Create CalendarDto, attach ValidationRules and unit tests (1 day)
c. Create ObjectPaths for CalendarDto and unit tests (1 day)
d. Create CalendarDtoFactory and unit tests (1 day)
e. Create JSP (2 days)
f. Update Controller servlet and unit tests (3 days) Assuming sorting, filtering functionalities etc to be implemented in the servlet

Server Side (10.5 days)

a. Create AddCalendarServiceRequest, CalendarServiceEvent and CalendarServiceProvider and unit tests – Servlet tier. (1 day)

b. Create Calendar domain object (1 day)
c. Create CalendarConverter that converts domain to dto and back and forth (1 day)
d. Create CalendarBusinessDelegate and implement createCalendarSchedule and unit test (0.5 days)
e. Create InMemoryCalendarRepository, implement createCalenderSchedule and unit tests (1 day)
f. Create PersistentCalendarRepository, implement createCalenderSchedule and unit tests (1.5 days)
g. Create CalendarManagerBean. Implement createCalenderSchedule with business validation and unit tests (2 days)
h. Create Calendar entity bean and map entity bean to the database schema (1 day)
i. Unit test for entity bean(1 day)
j. Add newly added beans to ServiceLocator (0.5 day)

READ (19.5 days) Same amount of work required as CREATE
UPDATE (19.5 days) Same amount of work required as CREATE
DELETE (19.5 days) Same amount of work required as CREATE

Add a new createCalenderSchedule functionality using Webservice

CREATE (24.5 days)

WebService Layer (14 days)

a. Model and create XML schema's for Value objects (3 days)
b. Model and create XML schema's for Exception classes (1 day)
c. Create WSDL (3 days)
d. Generate java stub classes from WSDL (1 day)
e. Create CalendarWebserviceDelegate class. Implement createCalendarSchedule and unit tests (3 days)
f. Implement createSchedule in CaleandarWebserviceImpl(1 day)
g. Update global deployment descriptor and unit tests (2 days)

Server Side (10.5 days)

a. Create AddCalendarServiceRequest, CalendarServiceEvent and CalendarServiceProvider and unit tests – Servlet tier. (1 day)
b. Create Calendar domain object (1 day)
c. Create CalendarConverter that converts domain to dto and back and forth (1 day)
d. Create CalendarBusinessDelegate and implement createCalendarSchedule and unit test (0.5 days)
e. Create InMemoryCalendarRepository, implement createCalenderSchedule and unit tests (1 day)
f. Create PersistentCalendarRepository, implement createCalenderSchedule and unit tests (1.5 days)
g. Create CalendarManagerBean. Implement createCalenderSchedule with business validation and unit tests (2 days)
h. Create Calendar entity bean and map entity bean to the database schema (1 day)
i. Unit test for entity bean(1 day)
j. Add newly added beans to ServiceLocator (0.5 day)

READ (24.5 days) Same amount of work required as CREATE
UPDATE (24.5 days) Same amount of work required as CREATE
DELETE (24.5 days) Same amount of work required as CREATE

Estimated days to add a brand new create functionality for each application

JIS NG

CREATE (11.5 days)
Client side (6 days)
Create View(3 day)
Create Controller and unit tests (3 days)

Server Side(5.5 days)
Create Calendar domain object, and validation rules and unit tests (2.0 day)
Create CalendarBusinessDelegate and implement createCalendarSchedule and unit test (0.5 days)
Create CalendarManagerJiniService, add this to ServiceLocator.. Implement createCalenderSchedule with business validation and unit tests (2 days)
Create Hibernate map to the database schema, and unit tests (1 day)
READ (11.5 days) Same amount of work required as CREATE
UPDATE (11.5 days) Same amount of work required as CREATE
DELETE (11.5 days) Same amount of work required as CREATE

Add a new createCalenderSchedule functionality using JSF WEB UI
CREATE (11.0 days)
Client side (5.5 days)
a. Create JSF (2 days)
b. Create BackingBeans and unit tests.(2.5 day)
c. Create Application configuration resource files. (1.0 day)

Server Side (5.5 days)
Create Calendar domain object, and validation rules and unit tests (2.0 day)
Create CalendarBusinessDelegate and implement createCalendarSchedule and unit test (0.5 days)
Create CalendarManagerJiniService, add this to ServiceLocator.. Implement createCalenderSchedule with business validation and unit tests (2 days)
Create Hibernate map to the database schema, and unit tests (1 day)
READ (11.0 days) Same amount of work required as CREATE
UPDATE (11.0 days) Same amount of work required as CREATE
DELETE (11.0 days) Same amount of work required as CREATE

Add a new createCalenderSchedule functionality using Webservice

CREATE (19.5 days)

WebService Layer (14 days)

Model and create XML schema's for Value objects (3 days)
Model and create XML schema's for Exception classes (1 day)
Create WSDL (3 days)
Generate java stub classes from WSDL (1 day)
Create CalendarWebserviceDelegate class. Implement createCalendarSchedule and unit tests (3 days)
Implement createSchedule in CaleandarWebserviceImpl(1 day)
Update global deployment descriptor and unit tests (2 days)

Server Side(5.5 days)

Create Calendar domain object, and validation rules and unit tests (2.0 day)
Create CalendarBusinessDelegate and implement createCalendarSchedule and unit test (0.5 days)
Create CalendarManagerJiniService, add this to ServiceLocator.. Implement createCalenderSchedule with business validation and unit tests (2 days)
Create Hibernate map to the database schema, and unit tests (1 day)
READ (19.5 days) Same amount of work required as CREATE
UPDATE (19.5 days) Same amount of work required as CREATE
DELETE (19.5 days) Same amount of work required as CREATE

Estimated days to add a new field (maintenance) in the existing domain model

ACORDS

Using SWING UI (9 days)
Client side (2 days)

a) ValueObject (0.25 day)
    - Add new property to a value objects
b) CalendarDetail (0.5 days)
    - Add new View Component to Layout screen
c) CalenderView (1.25 days)
    - Applying validation rules to the new property(0.5 day)
    - Register UI component with ViewSupport and unit test (0.25 day)
    - Test Pack/Unpack by runing unit tests (0.5 days)

Server side (7 days)

a) CalendarManagerBean – SessionBean (6 days)
    - Implement business validation for the new attribute and unit tests. Note: Estimation considers the fact that this is a huge class which is not refactored (1.5 day).
    - Manually locate calls to this service and verify it does not break any other service. Note: This step is required because current unit test suit does not provide a full coverage (4 days)
    - Update the SQLs which retrieve data (0.5 day)
b) Calendar - Entity bean (1 day)
    - Update EntityBean mapping to map to the new field(0.5 days)
    - Unit test for entity bean (0.5 day)

Estimated days to add a new field (maintenance) in the existing domain model

CAPS

Using Web UI (3.75 days)

1) Client Layer (1.75 days)
    a) CalendarValueObject (0.25 day)
    b) CalendarHelper (0.5 days)
        - Applying validation rules to the new property. Processes the action and handles error conditions.
    c) Calendar.jsp (1 day)

2) Server Layer (1 days)
    a) CalendarManagerBean (1 day)
        - Implement business validation and error messaging.
3) Persistence/DAO Layer (1 days)
    a) CalendarQueryEngine (0.5 days)
        - JDBC part to obtain connection, Create and Execute a Statement, close connection etc.
    b) CalendarSQLHelper (0.5 days)
        - The insert SQL for Calendar create.

Estimated days to add a new field (maintenance) in the existing domain model

JIS

Using SWING UI (6 days)

Client side (2.5 days)

a. Create new ValidationRule if required (0.25 day)
b. Create CalendarDto, attach ValidationRules and unit tests (0.25 day)
c. Create ObjectPaths for CalendarDto and unit tests (0.25 day)
d. Update View interface (0.25 day)
e. Update GUIView ,TestView and unit tests (0.5 days)
f. Update Controller and unit tests (1 days)

Server Side (3.5 days)

a. Update Calendar domain object (0.25 day)
b. Update CalendarConverter that converts domain to dto and back and forth (0.25 day)
c. Update InMemoryCalendarRepository, implement createCalenderSchedule and unit tests (0.5 day)
d. Update PersistentCalendarRepository, implement createCalenderSchedule and unit tests (1 day)
e. Update CalendarManagerBean for business validation and unit tests (0.5 days)
f. Update Calendar entity bean and map entity bean to the database schema (0.5 day)
g. Unit test for entity bean(0.5 day)

Estimated days to add a new field (maintenance) in the existing domain model

JIS NG

Using SWING UI (2.5 days)

Client side (1.0 days)
- Update View and unit tests(0.5 day)
- Update Controller and unit tests (0.5 days)

Server Side (1.5 days)
- Update Calendar domain object and unit tests (0.5 day)
- Update CalendarManagerJiniService for business validation and unit tests (0.5 days)
- Update Hibernate mapping to the database schema (0.5 day)

Estimated days to make a major change on UI and business rules for an existing module

Enhancement title: If user tries to schedule an anchor case for hearing, schedule the consolidated cases also on the same day. Show a pop up window showing the cases that are consolidated and the time slots available on that day. User selects the cases and the time slot from the pop up. Application schedules all these cases for hearing on the same day on the specified time slots and shows them on the screen.

ACORDS

Client side (7 days)

a) ValueObject (1 day)
   - Change the ValueObject to include multiple case numbers (enumeration) and unit tests (1 day)

b) Create CalendarConsolidatedCasesConfirmationWindow (1.5 day)

c) CalenderView (4.5 days)
   - Applying validation rules for the popup window and unit tests (0.5 days)
   - Perform actions to call business delegate to get the consolidated cases and unit tests (1 day)
   - Register PopupWindow UI components with ViewSupport and unit tests (1 day)
   - Pack/Unpack data for the popup window and unit tests (1 day)
   - Update BusinessDegate(RMIClientProxy) interface, implementation and unit tests for enhancing the service retrieveTimeSlots to accept a date parameter so that it can retrieve the available time slots for a given day (0.5 days)
   - Update BusinessDegate(RMIClientProxy) interface, implementation and unit tests for adding the new service getConsolidatedCases (0.5 days).

Server side (11 days)

a) RMI Layer (1 day)
   - Update RMIServlet interface, implementation and unit tests for enhancing retrieveTimeSlots() (0.5 days)
   - Update RMIServlet interface, implementation and unit tests for enhancing getConsolidatedCases() (0.5 days)

b) Façade (1 day) for enhancing retriveTimeSlots
   - Update ServerProxyCMPBean session bean interface, implementation and unit tests for enhancing retriveTimeSlots. This class is very fragile and too big to handle with 9000 lines of code (0.5 day)
   - Update ServerProxyCMPBean session bean interface, implementation and unit tests for adding the new service getConsolidatedCases. (0.5 days)

c) CalendarManagerBean – SessionBean (6 days)
   - Update business validation for retriveTimeSlots and unit tests. (1 day)
   - Update business validation for createCalendarSchedule and unit tests. This service should now accept an enumeration of case numbers (1 day)
   - Enhance createCalendar functionality and unit tests(2 days)

- Enhance retrieveTimeslots functionality and unit tests(2 days)

d) CaseManagerBean (3 days)
- Add business rules for new service retrieveConsolidatedCases (1 day)
- Add new service retrieveConsolidatedCases and unit tests(2 days)

CAPS

1) Client Layer (5 days)
   a) CalendarValueObject (1 day) –
      - Change the ValueObject to include multiple case numbers (enumeration) and unit tests (1 day)
   b) CalendarConsolidatedCasesConfirmationWindow Helper (2.0 days)
      - The controller class which has the implementation for the "CalendarConsolidatedCasesConfirmationWindow" action. Processes the action and handles error conditions.
   c) CapsWebHelperFactory (0.5 day)
      - Update this to create/retrieve an instance of required Helper class
   d) CalendarConsolidatedCasesConfirmationWindow.jsp (1 day)
   e) ClientProxy (0.5 day)
      - Update this to have the "getConsolidatedCases" and "retrieveTimeSlots" method pass-through

2) Server Layer (2.5 days)
   a) SessionProxyManager (0.5 day)
      - Update the façade to delegate the "getConsolidatedCases" and "retrieveTimeSlots" method call to the correct Session bean using Service Locator.
   b) CalendarManagerBean (1 day)
      - Implement business validation and error messaging.
   c) CaseManagerBean (1 day)
      - Implement business validation and error messaging
      - Create "getConsolidateCases" method and unittests.
3) Persistence/DAO Layer (4 days)
   a) CalendarQueryEngine (1 days)
      - Implement the method for "retrieveTimeSlots", i.e. do the JDBC part to obtain connection, Create and Execute a Statement, close connection etc.
   b) CalendarSQLHelper (1 day)
      - The SQL for "retrieveTimeSlots"
   c) CaseQueryEngine (1 days)
      - Implement the method for "getConsolidatedCases", i.e. do the JDBC part to obtain connection, Create and Execute a Statement, close connection etc.
   d) CaseSQLHelper (1 day)
      - The SQL for "getConsolidateCases"

JIS

Client side (8 days)

a. Create new ValidationRule if required (0.5 day)
b. Update CalendarDto and CalendarObjectPath to contain the list of consolidated cases (0.5 day).
c. Create CalendarConsolidatedCaseDto, attach ValidationRules and unit tests (0.5 day)
d. Create ObjectPaths for CalendarConsolidatedCaseDto and unit tests (0.5 day)
e. Create CalendarConsolidatedCaseDto Factory and unit tests (0.5 day)
f. Create View interface (0.5 day)
g. Create GUIView ,TestView and unit tests (2 days)
h. Create Controller, Action classes, and unit tests (2 days)
i. Update JISMainController to incorporate new controller (1 day).

Server Side (10 days)

a. Create CalendarConsolidatedCaseServiceRequest, CalendarConsolidatedCaseServiceEvent and CalendarConsolidatedCaseServiceProvider and unit tests – Servlet tier. (1 day)
b. Create CalendarConsolidatedCase domain object (0.5 day)
c. Update Calendar domain object to contain the list of consolidated cases (0.5 day).
d. Create CalendarConsolidatedCaseConverter that converts domain to dto and back and forth (0.5 day)
e. Create getConsolidatedCases in CaseBusinessDelegate and unit tests (0.5 days)
f. Update InMemoryCaseRepository to implement getConsolidatedCases (1 day).
g. Update PersistentCaseRepository to implement getConsolidatedCases (1 day).
h. Update CaseManagerBean to implement getConsolidatedCases and unit tests (1 day)
i. Update CalendarConverter to handle the list of consolidated cases (0.5 day).
j. Update CalendarBusinessDelegate to handle the list of consolidated cases and unit test (0.5 days)
k. Update InMemoryCalendarRepository to handle the list of consolidated cases and unit test (1 day)
l. Update PersistentCalendarRepository to handle the list of consolidated cases and unit tests (1.5 days)
m. Update CalendarManagerBean to handle the list of consolidated cases with business validation and unit tests (0.5 day).

JIS NG (9 days)

Client side (4.0 days)

a. Create View and unit test(1.5 day)
b. Create Controller, Action classes, and unit tests (2.5 days)

Server Side (5 days)

a. Create CalendarConsolidatedCase domain object (0.5 day)
b. Update Calendar domain object to contain the list of consolidated cases (0.5 day).
c. Create getConsolidatedCases in CaseBusinessDelegate and unit tests (0.5 days)
d. Update CaseManagerJiniService to implement getConsolidatedCases and unit tests (1 day)
e. Update CalendarBusinessDelegate to handle the list of consolidated cases and unit test (0.5 days)
f. Update CalendarManagerJiniService to handle the list of consolidated cases with business validation and unit tests. This service should now accept an enumeration of case numbers (1.0 day).
g. Enhance retrieveTimeslots functionality and unit tests(1 days)

APPENDIX F: JIS NG ARCHITECTURE

Introduction
JIS NG is the tag name for the future or Next Generation of the architecture that will host all new development, as well as, the migration of existing AOC applications. These recommendations are intended to address the limitations of the current AOC architecture as described in **Appendix C.**

Several important issues need to be resolved by a comprehensive architecture:
1) Producing the most productive development environment possible while satisfying the needs of the business.
2) Keeping the developer learning curve and training requirements to a minimum.
3) Selecting tools and technologies that have longevity and are standardized or have source code available.
4) Carefully weighing cost to develop or purchase against perceived benefit.
5) Providing the minimum of complexity that satisfied the expected application requirements.
6) Balance speed of development against deployments needs such as scalability and performance.

Architectural Layers and Technology Options
Architectures are typically broken into layers. Layers are supposed to limit dependencies by only allowing one layer to talk with the layer directly above and the layer directly below. Limiting dependencies yields the freedom to change the technology used in one layer while minimizing the impact to other layers. One technology or tool may provide the functionality of multiple layers, but ideally each layer is independent and can be swapped-out or optimized without affecting other layers.
Database layer

A developer only database is needed that is kept separate from all production style databases. As late as possible, the development database schema on new projects should be altered to match the expected production database. This will allow development to proceed at the fastest speed possible. Having an Object/Relational mapping layer is critical to allowing this flexibility.

The database architecture must be improved to provide a simplified interface for the Object/Relational mapping layer. This is necessary to provide adequate run-time performance and to ensure that the productivity of the application developer is optimal.

Options:
DB2

Recommendation
DB2 - There is no technical reason to switch from DB2.

Persistence layer
Persistence, Object/Relational mapping, and transactions are all addressed in this layer.

Data access implementation strategy
While JDBC is a given with Java systems it is best to have no hard coded SQL anywhere in the application and no stored procedures and limited use of database integrity rule implementation. Any database-level issues that are exposed in the applications code or rules that are captured in the database will slow down the speed at which application code can be developed and changed.

Mapping layer

A mapping layer cleanly separates the application code from the database. Also the mapping allows the application and database to grow and change independently, which allows DBAs to make changes without too much concern to the impact on applications so less coordination between the groups and less reconciling of their divergent concerns is required.

Options:

Hibernate
Hibernate is a popular Java Object/Relational mapping solution.

Java Data Objects (JDO) version 2
JDO is a standard with both open and commercial options available.

TopLink
TopLink is an Oracle product that supports DB2.

Recommendation
Hibernate is strongly recommended as an Object/Relational Mapping solution. It is the future of object relational mapping and is highly recommended by developers that are using it.

Hibernate is the most popular object/relational mapping solution for Java, with thousands of deployed systems, an active community of 6000 registered users all over the world, and tens of thousands of developers working with Hibernate day to day.

While looking for proven relational persistence solutions, the EJB3 Expert Group was inspired by this success and picked several key features of Hibernate for inclusion in the next major version of the EJB industry standard. We are very happy to see this first industry-wide effort to integrate Full Object/Relational Mapping into a central Java specification.

The EJB3 specification will support transparent persistence of plain Java objects, with a very similar feature set to Hibernate. The EntityManager and Query interfaces are also similar. This means that you will be able to use your Hibernate knowledge when creating EJB3 applications. You will find yourself in a familiar environment with outstanding object/relational mapping features, focused on the best possible integration with relational databases.

Domain layer
The domain layer contains the application's business logic.

Options:

Simple Java objects built with the JavaBean standards.

Recommendation

Simple Java objects - Sticking with simple Java objects, as much as possible, speeds-up development and improves overall quality.

Data Transfer Layer

The data transfer layer provides the mechanism for transferring domain layer data to clients.

Options:

Metadata-driven user interface
Strategies exist which allow user interfaces to be generated directly from metadata which describe their relationship with the domain layer. The user interface components can be bound automatically to the domain layer data.

Automatic mapping of domain layer to transport format
The domain layer data will be sent to the client using a two step process. The data will be converted automatically to a serialization format for transport. Additionally, the client user interface components must be explicitly mapped to the expected transport data.

Recommendation
The metadata-driven approach would only be effective if standardization of the UI could be done without hand building and tweaking the UI could then directly using the domain model.
For the JIS NG architecture, it is recommended to use an Automatic mapping approach that standardizes distributing domain object to clients and minimizes hand coding.

Service Layer

The service layer exposes the domain layer's business services.

Options:

JINI services.
JINI is a Sun originated standard for service-based system development that is provided for free with source.

EJB Session Beans
EJB Session Beans can provide security and transaction management but require complicated development and deployment processes.

Recommendation
JINI is the best way to deal with distributed object systems that are service-oriented and is recommended over Session beans due to JINI’s lower overhead and speedier development model.

JINI's use for Service Oriented Architectures
> excerpt from
http://www.computerworld.com/softwaretopics/software/appdev/story/0,10801,94945,00.html
AUGUST 02, 2004 (COMPUTERWORLD) - For some companies, procuring software isn't a binary buy-it-or-build-it proposition. "We are largely an open-source shop, so when we think about buying software, there's a general aversion to it," says Orbitz CTO Chris Hjelm. The online travel company uses open-source Linux on most of its 1,000 servers and the free JINI network architecture for distributed systems.

"We've built our services layer such that when you make a call to the JINI framework – to buy an airline ticket, for example – it manages that request. It goes off and finds all the back-end complex things that have to happen to build an airline ticket, and it makes that transparent," he says.

> excerpt from http://sys-con.com/story/?storyid=44361&DE=1
Webservices Journal - SOA Came to Boston at EDGE (East) 2004

Recounting briefly the history of Orbitz's service-oriented architecture, Hjelm said that Java was the first big decision, and JINI the second. "So when SOA became popular, Orbitz had already found it.

"The JINI distributed computing framework focuses on interfaces and capabilities not implementation and location," Hjelm said.
…
"If you were to take the average person and explain GDS to them, their head would hurt," he added, "whereas we can get developers up to speed on Orbitz fast. We add a new machine, bring it up into the network, and JINI recognizes it and starts to draw on it. It works.

"So the growth in our code base isn't in the services layer, it's in the application layer," he pointed out. That is the key to Orbitz's success, and that in turn is a function of its architectural choices.

Transport layer

The transport layer provides the protocol for sending data across the network.

Options:

JERI
JERI is Sun's latest RMI implementation that allows flexible implementation.

RMI/JRMP
Sun's RMI/JRMP is the original Remote Method Invocation mechanism for providing the ability to call Java methods on remote applications.

Web Services

All service clients, including Swing, access the business services through web services.

Recommendation
JERI provides more power than RMI while keeping the benefits of a Java optimized solution. JERI stands for JINI Extensible Remote Invocation, the Extensible part allows for options such as replacing the transport protocol with http allowing it to work across firewalls through the standard port 80 just like web services and web browsers.

JiniExtensible Remote Invocation is a new implementation of the Java Remote Method Invocation (RMI) programming model that provides APIs for customizing remote method invocation behavior on a per-remote object basis. The JINI ERI framework provides pluggable components representing the various layers of the RMI protocol stack; by extending or replacing these components, applications can tailor the transport, invocation and dispatch behavior of remote method calls.

Client layer
There are two types of clients that are needed, a Rich client for heavy data entry and a public Web client. It may be possible to support both sets of users with one type of UI technology be either distributing Swing over the web or by using a Rich Web client for both audiences. Additionally, a web service interface is required.

Rich Client with Swing

Options:

Metadata-driven UI generated with templates
Strategies exist which allow user interfaces to be generated automatically using templates and metadata mapping. No custom coding of Swing components is necessary. Swing components are dynamically created and automatically bound to domain data.

User interface components bound by tags to domain data
Object and attribute names will be used to manually tag UI components to allow binding between pre-built UIs and domain data.

Recommendation
If the users can be convinced to give up some control over the UI behavior of each window and accept a generalized approach, such as the Universal Navigation strategy, then there would be a large cost and time savings achieved by generating the UI dynamically instead of manually. There would not be any overall usability compromises, but any usability requirements would be implemented in a common way for all windows of a given task type and would not be customizable for a specific window.

If optimum usability is selected over cost and speed, then having another group build the windows and provide tags that connect them to the domain data is the best option.

It is also recommended that third party Swing components be used instead of writing custom components for such things as Calendar controls.

Public Web access

Options:

Java Server Faces (JSF)
Java Server Faces is a Java standard which provides a programming model similar to Swing for web application development.

Java Servlets with a template engine such as Velocity
Velocity is one of many template engines used for rendering web user interfaces.

Recommendation
JSF is recommended as it provides a model that allows much of the user interface code to be reused. A Swing client user interface can be exposed as a web interface with minimal effort.

Web service layer

Recommendation
Apache AXIS is recommended, along with the use of WSDL generation tools to speed up AXIS development.

Recommended Technology Stacks
The three types of user interfaces each need some unique technologies, but they all share as many solutions as possible.

Layer	**1) Web**	**2) Swing**	**3) Web service**
Client	JSF	Hand coded UI	AXIS
Transport	JERI	JERI	SOAP
Service	JINI	JINI	JINI
Transport format	Generic	generic	generic
Domain	JavaBeans	JavaBeans	JavaBeans
Persistence	Hibernate	Hibernate	Hibernate
Database	DB2	DB2	DB2

Deployment Environment
It is recommended that a JRE (Java Runtime Environment) version 5.0 be used for deployment. WebSphere is not required. It is recommended that a lighter weight web application container, such as Tomcat, be used.

To speed the development effort, any hurdles the developers face, i.e., deployment to Websphere in order to test each code change, should be removed. Tomcat will provide the most productive environment for developers.

Development Environment
To maximize the productivity of developers, it is important to have the flexibility to use the latest version of development tools. The most productive environment currently available is Eclipse 3.x plus JDK (Java Development Kit) 5.0. It is recommended that a current version of Eclipse be used with JDK 5.0.

The latest Java Runtime Environment (5.0) provides a number of developer productivity enhancements. Moving to the latest JDK at this point will lessen long-term costs as the training and effort required to move to the latest JDK can be rolled into the development effort. Many tools and technologies are starting to require JDK 5.0, which will start to limit the choices available and lessen the speed improvements possible even more as time progresses.

Programming Model

Metadata or Model-driven development
Metadata-driven development is a strategy that leverages configuration information kept separate from source code, allowing changes to an external file that will affect the application. These strategies save developer time by externalizing portions of the code that need to be frequently changed and need to be accessed from many places throughout the code.

Spring Container
Spring is a framework for simplifying configuration and lessening dependencies. The key benefit to this approach is to allow technologies to be switched out by changing external configuration files and not source code. Also it lessens the amount of technology specific Java code that developers have to write, keeping the code cleaner, simpler and easier to understand and maintain.

Security
It is recommended that a security architecture be integrated with a dependency injection framework such as Spring or PicoContainer. Acegi Security is a standard security approach when using such a framework.

Reporting
It is recommended that a third party reporting component be integrated into the JIS NG architecture. There are several available such as Jasper Reports, Style Reports, and JReport.

APPENDIX G: EVALUATED SHORT TERM ALTERNATIVES

These development efforts are fairly independent and in general could be pursued concurrently.

Deliver the JIS Migration April 2005 release goals
The goals which the JIS Migration was intended to support for the October 2004 and April 2005 release could be met by June 2005, as long as, an effort is taken to significantly improve the database and application architectures. If these JIS Migration goals are no longer desirable, it would also be possible to achieve other business goals of a similar scope. This would require a short effort of two to three months to improve the application and database architectures. After these improvements are in place, the development effort could be scaled up to support additional teams.

These changes will allow development to proceed at a much faster pace than was previously possible. To optimize this strategy, an increase in the amount of QA and DBA resources will be required.

Add CAPS scheduling functionality to ACORDS.
If it is desirable to provide ACORDS users with scheduling functionality that is similar to CAPS and is integrated into the ACORDS application, it is possible that this could be achieved in the June 2005 time frame. This new version of ACORDS would be identical to the current version but the scheduling functionality would be replaced.
This would provide the following benefits:

- Support a second court level. Appelate court users would have the required portion of the CAPS scheduling model seamlessly integrated into their application.
- Migration of scheduling functionality to a new architecture. This effort would involve the creation of a new Calendaring service, initially just to serve the ACORDS users, but later generalized for all users. This service could be more easily integrated in other applications and would be built on JIS NG.
- Seamless integration with ACORDS. The strategy is to merge the current ACORDS UI with the new CAPS Swing UI so that users do not realize that they are really accessing two backend systems.

Integration with external applications
There are several efforts which could be under taken to provide services that would integrate with systems external to AOC.

Some of the potential benefits include:

- Integration with document management systems.
- Data extraction efforts can be improved. A data extraction architecture could be put into place that will be reusable for multiple data extraction efforts. This would support sending defined datasets to external customers. For example, an external customer would setup for themselves the criteria of the datasets they require and the frequency that the

extractions are needed. These datasets would be automatically extracted based on the criteria and would be pushed or be made available for retrieval per the defined schedule.
- Current systems won't require modification. A new service layer could be added that allows the current systems to remain in place.
- Reusable service framework could be developed. As new services are put into place, a sophisticated service framework could be developed which would minimize the cost of future integrations. This interface would remain even as legacy systems are migrated to JIS NG and would shield external users from changes to internal applications.

Required Parallel Efforts
In order for any of the sort term goals to be successful two parallel efforts need to be undertaken, either at the same time as the short-term goal selected or as a precursor to the short-term goal.

Create new platform – JIS NG
Work should begin on an entirely new development platform. This new platform and architecture would be the foundation for all new application development, including new features for existing systems, as well as, legacy application migrations.

This would be a concentrated effort to create an optimal architecture designed to create a high-speed development environment that is quick to learn and makes use of standardized architectural components. The platform would assist AOC in achieving business goals with a minimal amount of time and using readily available resources, i.e., it will be designed to allow less experienced developers to be effective. The major benefit to the business is reduction in ongoing costs for new and maintenance development and more predictable and shorter timelines for all future development efforts.

Database architecture improvements
Currently, the database used by CAPS, ACCORDS, JIS, and other applications suffers from poor performance problems and is extremely difficult to develop new applications upon. Improvements need to be made to the database architecture to drastically reduce the cost of new application development and improve performance. Options include:

- Reduce table size. The current production database contains too much data in certain tables. An effort could be made to improve the performance and manageability of the database by archiving off older records or by partitioning existing large tables into multiple smaller tables.
- Developing new schemas with real-time synchronization to the legacy database. The data model must be evolved from the legacy mainframe structures if any new application architecture is going to be successful. There are several approaches that could support this goal. Application developers need new databases that have meaningful naming conventions and more closely model the new application domain. Any new databases will likely require a real-time synchronization with the legacy systems. There are several ways this could be achieved.
- Speed up development efforts. The complex database schema is a drag on development efforts and some approach needs to be decided upon to address development speed. Options include Object/Relational mapping, using views to consolidate tables and database schema simplification.

APPENDIX H: REQUIRED DEVELOPMENT SUPPORT TEAMS

Roles and Responsibilities

One important aspect of speeding-up development is to focus the development team on business coding and limit their need to switch out of writing object code and into writing UI, architecture or database code. This type of context switching wastes development time and these separate areas are best handled by team members with the appropriate expertise. The following is a list of the teams and their responsibilities that are needed to keep the developers focused on realizing new business behavior and away from being sidetracked by tangential issues.

1) UI Behavior Team

- Work with users to design the UI.
- Build all UIs for both Swing and Web.
- Hand-off to dev team to add logic and to tie UI into the backend.
- The UI implementations need to stay one iteration ahead of development so that development is never waiting for the UI to be built.
- Will coordinate with development to ensure that suggested UI designs are feasible to implement.

2) Development DBAs

- Coordinate with Enterprise DBAs to support the needs of the Dev Team.
- Handle mapping domain objects to the database schema.
- Help write complex queries to off-load these tasks, and the knowledge required to write them, from the developers.
- Optimize schema, queries and object mappings for performance.
- Involved with development in design discussions to raise issues and to implement any changes to the mapping or database layers.
- Organize data and databases for unit testing and development prototyping.
- Work with Enterprise DBAs to push schema changes into test and productional environments.

3) Testing Team

- Write user-oriented tests that prove that the needed business functionality has been implemented.
- Run functional tests to ensure that previously correct behavior remains intact, i.e., regression testing.
- Also charged with other forms of testing such as load and performance testing.
- Will create sets of test data that reflect business reality and assist in testing edge conditions in both unit and functional tests.
- Should be able to write system and integration tests using Java and not just visual tests with a fancy tool.

4) Business Analysis Team

- Determine business needs.
- Work to consolidate the business needs into requirements that the dev team can implement.

- Stay at least one iteration ahead of development so that development is never waiting for work.
- Available for daily questions about requirements during implementation.
- Approve development deliverables.
- Work with the testing team to write test plans and review functional tests.
- Work with the UI team to create UI designs and do preliminary usability testing.

5) Architecture Team

- This team may include team leads and head designers along with pure architects.
- Mentor dev team members on proper use of the existing architecture, processes and tools.
- Determine where the architecture can be enhanced to improve development speed.
- Implement architectural enhancements and train development on any changes.
- Review available technologies, processes and techniques and suggest options that should improve development quality and speed.
- Review code to provide feedback on correct use of the architecture and to improve code quality.
- Build, find or buy tools that assist development and testing of components built on the architecture.
- Review upcoming UI and Business needs to determine the fit/gap with the current architecture and suggest alternatives that match the architecture and/or prepare to fill the gaps with new architecture or tools.
- Participate in daily development team meeting, project planning and iteration retrospectives to assist and support the development team.
- Be available to provide guidance to developers and assist in coding difficult or new types of development tasks.

APPENDIX I: SOLUTIONSIQ ENGAGEMENT CONTACTS

Table 1: Project Contact Information

	First Contact: Enterprise Solution Manager*	**Second Contact:** Primary Technical Contact	**Third contact:** Secondary Technical Contact
Name	Julia Francis	Will Iverson	Evan Campbell
Title	Enterprise Solutions Manager	Practice Manager	CTO/VP of Professional Services
Phone	425.519.6718	425.451.2727 x2562	425.451.2727
Email	JFrancis@SolutionsIQ.com	WIverson@SolutionsIQ.com	ECampbell@SolutionsIQ.com

APPENDIX J: DOCUMENT HISTORY

Table 2: Document History

Date	Version #	Owner	Section	Modification